TI-Nspire™ program Editor GuideBook

Få mere at vide om TI Technology via online-hjælpen på education.ti.com/eguide.

Vigtige oplysninger

Medmindre andet udtrykkeligt angives i den Licens, der følger med et program, stiller Texas Instruments ingen garantier, hverken udtrykkeligt eller underforstået, herunder men ikke begrænset til underforståede garantier om salgbarhed og egnethed til et bestemt formål, for programmer eller skriftligt materiale, og Texas Instruments stiller udelukkende sådant materiale til rådighed, som det foreligger. Texas Instruments kan under ingen omstændigheder holdes ansvarlig for særlige, indirekte, hændelige eller følgeskader i forbindelse med eller som følge af køb eller brug af dette materiale, og hele Texas Instruments' erstatningsansvar kan, uanset søgsmålets art, ikke overstige det beløb, der fremgår af programlicensen. Derudover kan Texas Instruments ikke holdes ansvarlig for nogen form for krav som følge af en anden parts brug af dette materiale.

TI-Nspire™ software bruger Lua som scriptingmiljø. For oplysninger om ophavsret og licens, se http://www.lua.org/license.html.

TI-Nspire™ software bruger Chipmunk Fysik version 5.3.4 i simuleringsmiljøet. information om licenser, se http://chipmunk-physics.net/release/Chipmunk-5.x/Chipmunk-5.3.4-Docs/.

Microsoft® og Windows® er registrerede varemærker tilhørende Microsoft Corporation i USA og/eller andre lande.

Mac OS®, iPad® og Mac OS X® er registrerede varemærker tilhørende Apple Inc.

Unicode® er et registreret varemærke tilhørende Unicode, Inc. i USA og andre lande.

De faktiske produkter kan variere let fra de viste billeder.

Indhold

Kom godt i gang med programeditoren

Du kan oprette brugerdefinerede funktioner eller programmer ved at skrive programsætninger i indtastningslinjen i Beregninger eller ved hjælp af programeditoren. Programeditoren har nogle fordele, og de beskrives i dette afsnit. For yderligere oplysninger henvises der til *Beregninger*.

- Editoren har programmeringsskabeloner og dialogbokse, som kan hjælpe dig med at oprette funktioner og programmer med korrekt syntaks.
- Med editoren kan du indtaste programsætninger på flere linjer uden at skulle bruge en speciel tastesekvens til at tilføje hver enkelt linje.
- Du kan nemt oprette private og offentlige biblioteksobjekter (variabler, funktioner og programmer). For yderligere oplysninger henvises der til *Biblioteker*.

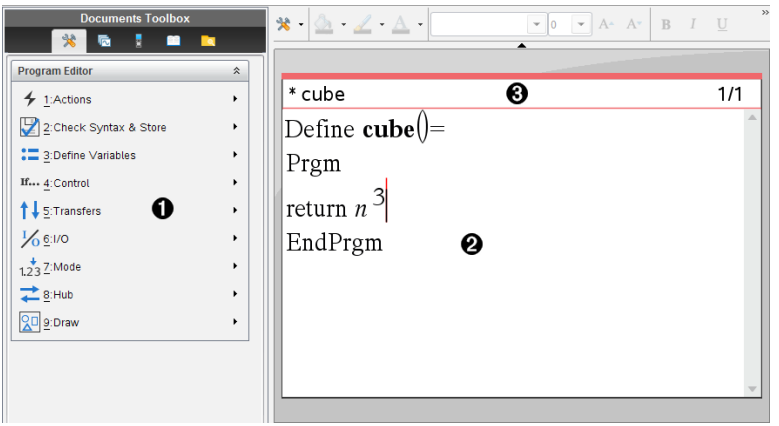
Åbning af programeditoren

▶ For at tilføje en ny programeditor-side i den aktuelle opgave:

På værktøjslinjen, klik **Indsæt > Programeditor> Ny**.

Håndholdt: Tryk på doc, og vælg **Indsæt > Programeditor > Ny**.

Bemærk: Editoren er også tilgængelig fra menuen **Funktioner & Programmer** på en Beregninger-side.



❶ Menuen i programeditoren er altid tilgængelig, når du er i programeditorens arbejdsområde og bruger normalvisning.

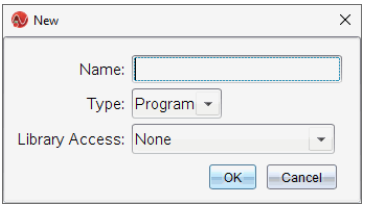
❷ Programeditorens arbejdsområde.

❸ Statuslinjen viser linjenummeret og navnet på den funktion eller det program, der redigeres. En stjerne (*) viser, at denne funktion er "snavset," hvilket vil sige, at den er ændret siden sidste gang, syntaksen blev kontrolleret, og den blev gemt.

Definering af et program eller en funktion

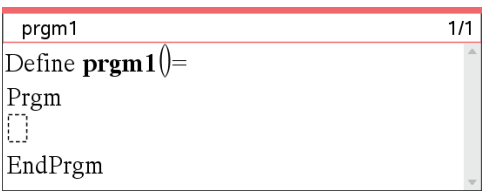
Start af en ny programeditor

1. Sørg for, at du er i dokumentet og opgaven, hvor du vil oprette programmet eller funktionen.
2. Klik på **Insert**-knappen på applikationens værktøjslinje, og vælg **Program Editor > New**. (På den håndholdte, tryk på doc og vælg **Insert > Program Editor > New**.)



3. Skriv et navn på den funktion eller det program, du definerer.
4. Vælg **Type** (**Program** eller **Function**).
5. Indstil **Biblioteksadgang**:
   - Hvis du kun vil anvende funktionen eller programmet i det aktuelle dokument og opgave, skal du vælge **None**.
   - Hvis funktionen eller programmet skal være tilgængeligt fra alle dokumenter, men ikke synligt i kataloget, skal du vælge **LibPriv**.
   - Hvis funktionen eller programmet skal være tilgængeligt fra alle dokumenter, og også synlig i kataloget, skal du også vælge **LibPub (Show in Catalog)**. For yderligere oplysninger henvises der til *Biblioteker*.
6. Klik på **OK**.

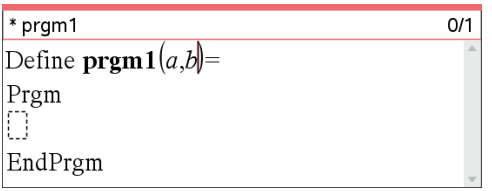
   En ny kørsel af Programeditoren åbnes med en skabelon, der passer til de valg, du har truffet.



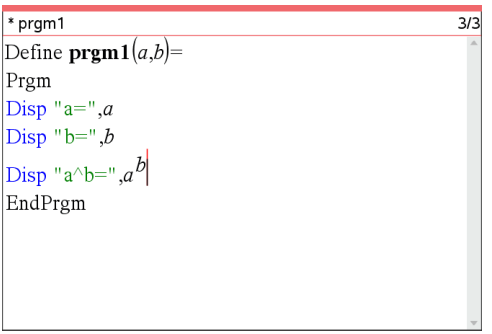
Indsætning af programlinjer i en funktion eller et program

Programeditoren udfører ikke kommandoerne eller beregningerne, i takt med at du har indtastet dem. De udføres først, når du kalder funktionen eller kører programmet.

1. Hvis din funktion eller program kræver, at brugeren indsætter argumenter, skal parameternavnene indsættes i de parenteser, der efterfølger navnet. Adskil parametrene kommaer.



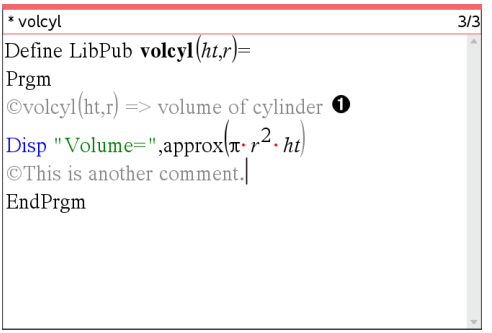
2. Indsæt de sætningslinjer, der udgør din funktion eller dit program mellem, Func og EndFunc (eller Prgm og EndPrgm)-linjerne.



- Du kan enten skrive navnene på funktionerne og kommandoerne eller indsætte dem fra kataloget.
- En linje kan ikke være længere end skærmens bredde; så du har muligvis brug for at rulle ned og op for at få vist hele sætningen.
- Når hver enkelt linje er skrevet, skal du trykke på **Enter**. Dette indsætter en tom linje, så du kan fortsætte med at indtaste en linje mere.
- Anvend piltasterne ◄, ►, ▲, og ▼ til at rulle gennem funktionen eller programmet og indtaste eller redigere kommandoer.

Indsætte kommentarer

Kommentarer kan være nyttige for dem, der gennemser eller redigerer programmet. De vises ikke, når programmet kører, og de påvirker ikke programafviklingen. Symbolet © vises i begyndelsen af linjen med kommentaren.



❶ Kommentar, der viser den påkrævede syntaks. Da dette biblioteksobjekt er offentligt, og denne kommentar er første linje i en Func- eller Prgm-blok, vises kommentaren i kataloget som hjælp. For yderligere oplysninger henvises der til *Biblioteker*.

Sådan indsættes en kommentar:

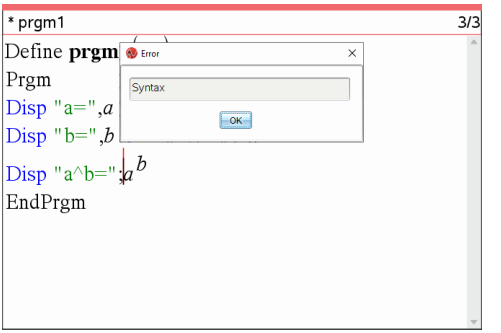
1. Placer markøren ved slutningen af den linje hvor du vil indsætte en kommentar.
2. Fra menuen **Actions**, klik på **Insert Comment**, eller tryk **Ctrl+T**.
3. Skriv teksten i kommentaren efter ©-tegnet.

Kontrol af syntaks

I Programredaktør kan du kontrollere, at funktionen eller programmet har korrekt syntaks.

- Fra menuen **Check Syntax & Store**, klik på **Check Syntax**.

  Hvis syntakskontrollen finder syntaksfejl, viser den en fejlmeddelelse, og prøver at placere markøren tæt på den første fejl, så du kan rette den.



Lagring af funktionen eller programmet

Du skal lagre din funktion eller dit program for at gøre dem tilgængelige. Programeditoren kontrollerer automatisk syntaksen før lagring.

Der vises en stjerne (*) i øverste venstre hjørne af Programeditoren for at vise, at funktionen eller programmet ikke er lagret.

- Fra menuen **Check Syntax & Store**, klik på **Check Syntax & Store**.

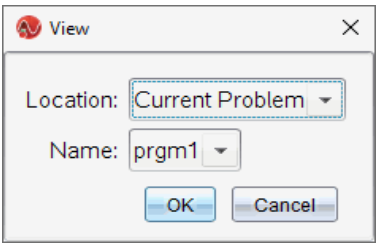
  Hvis syntakskontrollen finder syntaksfejl, viser den en fejlmeddelelse og prøver at placere markøren tæt på den første fejl.

  Hvis der ikke er fundet syntaksfejl, vil meddelelsen "Gemt succesfuldt" ("Stored successfully") blive vist på statuslinjen, i toppen af Programeditoren.

Bemærk: Hvis funktionen eller programmet er defineret som et biblioteksobjekt, skal du også gemme dokumentet i den dertil beregnede biblioteksmappe og opdatere bibliotekerne for at gøre objektet tilgængeligt for andre dokumenter. For yderligere oplysninger henvises der til *Biblioteker*.

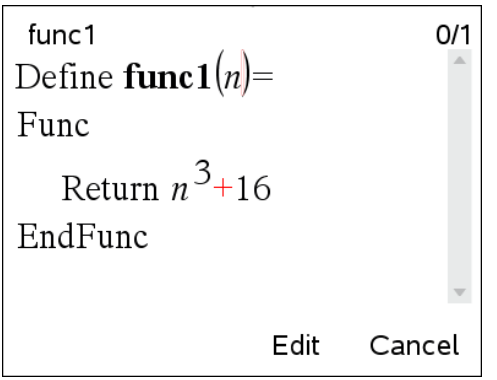
Visning af et program eller en funktion

1. Åbn menuen **Handlinger**, og vælg **Vis**.



2. Hvis funktionen eller programmet er et biblioteksobjekt, vælges dets bibliotek i listen **Placering (Location)**.
3. Vælg funktions- eller programnavnet i listen **Navn (Name)**.

   Funktionen eller programmet vises i en fremviser.



4. Brug piltasterne til at se funktionen eller programmet.

5. Hvis du vil redigere programmet, skal du klikke på **Rediger**.

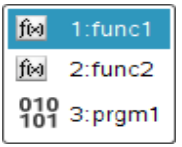
   Bemærk: Samlingen af værktøjer i **Rediger (Edit)** er kun tilgængelig for funktioner og programmer, der er defineret i denne aktuelle opgave. For at redigere et biblioteksobjekt skal du først åbne dets biblioteksdokument.

Åbning af en funktion eller et program til redigering

Du kan kun åbne en funktion eller et program i den aktuelle opgave.

Bemærk: Du kan ikke ændre låste programmer eller funktioner. Du kan låse objektet op ved at gå til en Regner-side og bruge kommandoen **unLock**.

1. Vis en liste med tilgængelige funktioner og programmer
   - Åbn menuen **Handlinger**, og vælg **Åbn**.

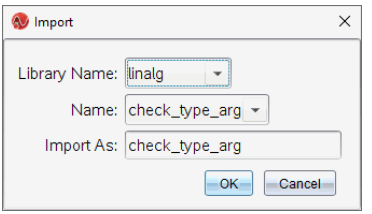


2. Vælg det element, der skal åbnes.

Import af et program fra et bibliotek

Du kan importere en funktion eller et program, der er defineret som et biblioteksobjekt til en programeditor i den aktuelle opgave. Den importerede kopi er ikke låst, heller ikke hvis originalen er låst.

1. Åbn menuen **Handlinger**, og vælg **Importer**.



2. Vælg **Biblioteksnavn (Library Name)**.
3. Vælg **Navn (Name)** på objektet.
4. Hvis det importerede objekt skal have et andet navn, skal du skrive navnet under **Importer som (Import As)**.

Oprettelse af en kopi af en funktion eller et program

Ved oprettelse af en ny funktion eller et program kan du finde det nemmere at starte med en kopi af den aktuelle. Den kopi, du opretter, er ikke låst, heller ikke selvom originalen er låst.

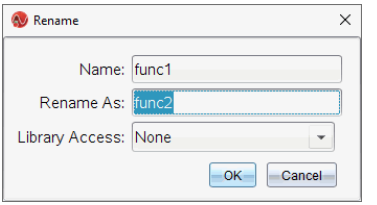
1. Åbn menuen **Handlinger**, og vælg **Opret kopi**.

2. Skriv et nyt navn, eller klik på **OK** for at acceptere det foreslåede navn.
3. Hvis du vil ændre adgangsniveau, skal du vælge **Biblioteksadgang (Library Access)** og vælge et nyt niveau.

Omdøbning af programmer eller funktioner

Du kan omdøbe og (valgfrit) ændre adgangsniveauet for den aktuelle funktion eller program.

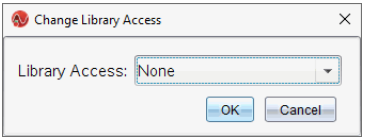
1. Åbn menuen **Handlinger**, og vælg **Omdøb**.



2. Skriv et nyt navn, eller klik på **OK** for at acceptere det foreslåede navn.
3. Hvis du vil ændre adgangsniveau, skal du vælge **Bibioteksadgang (Library Access)** og vælge et nyt niveau.

Ændring af biblioteksadgangsniveau

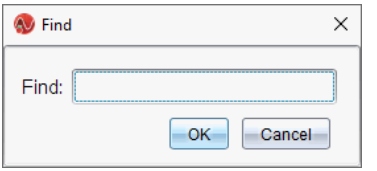
1. Åbn menuen **Handlinger**, og vælg **Skift biblioteksadgang** .



2. Vælg **Biblioteksadgang (Library Access)**:
   - Hvis du kun vil anvende funktionen eller programmet i den aktuelle opgave i Regner, skal du vælge **Intet (None)**.
   - Hvis funktionen eller programmet skal være tilgængeligt fra alle dokumenter men ikke i Katalog, skal du vælge **LibPriv**.
   - Hvis funktionen eller programmet skal være tilgængeligt fra ethvert dokument, også i Katalog, skal du også vælge **LibPub**.

Søgning i tekst

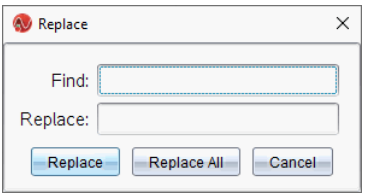
1. Åbn menuen **Handlinger**, og vælg **Søg**.



2. Skriv den tekst, du vil søge, og klik på **OK**.
   - Hvis teksten bliver fundet, fremhæves den i programmet.
   - Hvis teksten ikke bliver fundet, vises en meddelelse.

Søgning og erstatning af tekst

1. Åbn menuen **Handlinger**, og vælg **Søg og erstat**.



2. Skriv den tekst, du vil søge efter.
3. Skriv den tekst, der skal erstatte.
4. Klik på **Erstat** for at erstatte den første forekomst efter markørpositionen, eller klik på **Erstat alle** for at erstatte alle forekomster.

   Bemærk: Hvis teksten findes i en matematikskabelon, vises en meddelelse, der advarer om, at din erstatningstekst erstatter hele skabelonen, ikke blot den fundne tekst.

Lukning af den aktuelle funktion eller program

▸ Åbn menuen **Handlinger**, og vælg **Luk**.

Hvis funktionen eller programmet har ændringer, der ikke er gemt, bedes du kontrollere syntaksen og gemme, før du lukker.

Kørsel af programmer og beregning af funktioner

Du kan bruge det fra en applikation, efter at et program eller en funktion er defineret og gemt. Alle applikationer kan beregne funktioner, men kun Beregninger og Noter kan køre programmer.

Programsætningerne udføres i rækkefølge (selvom visse kommandoer ændrer programafviklingen). Resultatet, hvis der er et, vises i applikationens arbejdsområde.

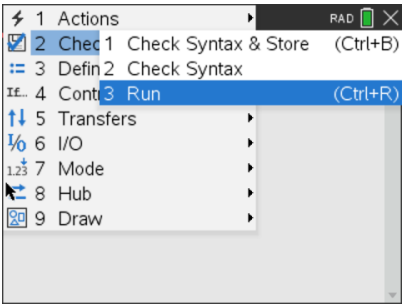
- Programudførelsen fortsætter, til den når den sidste sætning eller kommandoen **Stop**.
- Funktionsudførelsen fortsætter, til den når en **Return**-kommando.

Kørsel af et program eller funktion fra programeditoren

1. Sørg for at du har defineret et program eller funktion, og at programeditoren er det aktive panel (computer) eller side (håndholdt).
2. På værktøjslinjen, klik på **Dokumentværktøjer**-knappen  og vælg **Kontroller syntaks og gem > Kør**.

—eller—
Tryk på **Ctrl+R**.

Håndholdt: Tryk på menu 2 3, eller tryk på ctrl R.



Det vil automatisk:

- Kontrollere syntaksen og gemme programmet eller funktionen,
- Indsætte navnet på programmet eller funktionen på den første tilgængelige linje af beregningsapplikationen lige efter programeditoren. Hvis ingen Beregninger findes i den position, så sættes en ny ind.

3. Hvis programmet eller funktionen kræver, at du giver et eller flere argumenter, skal du skrive værdierne eller variabelnavnene inde mellem parenteserne.
4. Tryk enter.

Bemærk: Du kan også køre et program eller funktion i applikationerne Beregninger eller Noter, ved at skrive navnet på programmet med parenteser og de påkrævede argumenter, og trykke på enter.

Anvendelse af korte og lange navne

Når du er i den samme Opgave, hvor et objekt er defineret, kan du tilgå det ved at indsætte dets korte navn (navnet der vises i objektets **Define**-kommando). Dette er tilfældet for alle definerede objekter, inklusive private, offentlige og ikke-biblioteksobjekter.

Du kan tilgå et biblioteksobjekt fra ethvert dokument ved at skrive objektets lange navn. Et langt navn indeholder navnet på objektets biblioteksdokument, efterfulgt af en backslash "\", efterfulgt af navnet på objektet. Det lange navn på objektet defineret som **func1** i biblioteksdokumentet **lib1** er for eksempel **lib1\func1**. For at skrive "\" tegnet på den håndholdte, tryk [⇧shift] [÷].

Bemærk: Hvis du ikke kan huske det nøjagtige navn eller rækkefølgen på de krævede argumenter for et privat biblioteksobjekt, kan du åbne biblioteksdokumentet eller bruge programeditoren til at vise objektet. Du kan også bruge **getVarInfo** for at se en liste over objekter i et bibliotek.

Anvendelse af et offentligt biblioteksprogram eller funktion

1. Sørg for, at du har defineret objektet i dokumentets første opgave, gemt objektet, gemt biblioteksdokument i mappen MyLib og opdateret bibliotekerne.
2. Åbn den TINspire™-applikation, hvori du vil anvende programmet eller funktionen.

   Bemærk: Alle applikationer kan beregne funktioner, men kun applikationerne Beregninger og Noter kan køre programmer.
3. Åbn Katalog og anvend biblioteksfanen til at søge og indsætte objektet.
   —eller—
   Skriv navnet på objektet. Ved programmer eller funktioner skal navnet altid efterfølges af parenteser.

   ```
   libs2\func1()
   ```
4. Hvis programmet eller funktionen kræver, at du giver et eller flere argumenter, skal du skrive værdierne eller variabelnavnene inde mellem parenteserne.

   ```
   libs2\func1(34,potens)
   ```
5. Tryk [enter].

Anvendelse af et privat biblioteksprogram eller en funktion

For at kunne anvende et privat biblioteksobjekt skal du kende dets lange navn. Objektets lange navn, der er defineret som **func1** i biblioteksdokumentet **lib1** er for eksempel **lib1func1**.

Bemærk: Hvis du ikke kan huske det nøjagtige navn eller rækkefølgen på de krævede argumenter for et privat biblioteksobjekt, kan du åbne biblioteksdokumentet eller bruge programeditoren for at se objektet.

1. Sørg for, at du har defineret objektet i dokumentets første opgave, gemt objektet, gemt biblioteksdokument i mappen MyLib og opdateret bibliotekerne.
2. Åbn TINspire™-applikationen, hvor du vil anvende programmet eller funktionen.

   Bemærk: Alle applikationer kan beregne funktioner, men kun applikationerne Beregninger og Noter kan køre programmer.
3. Skriv navnet på objektet. Ved programmer eller funktioner skal navnet altid efterfølges af parenteser.

```
libs2\func1()
```

4. Hvis objektet kræver, at du giver et eller flere argumenter, skal du skrive værdierne eller variabelnavnene inde mellem parenteserne.

```
libs2\func1(34,potens)
```

5. Tryk enter.

Afbrydelse af et program eller en funktion, der kører

Mens et program eller en funktion kører, vises markøren ⏲.

- Afbrydelse af programmet eller funktionen
  - Windows®: Hold tasten **F12** nede, mens der gentagne gange trykkes på **Enter**.
  - Mac®: Hold tasten **F5** nede, mens der gentagne gange trykkes på **Enter**.
  - Håndholdt: Hold tasten on nede, mens der gentagne gange trykkes på enter.

  En besked bliver vist. For at redigere programmet eller funktionen i programeditoren, vælg **Gå til**. Markøren vises ved kommandoen, hvor afbrydelsen opstod.

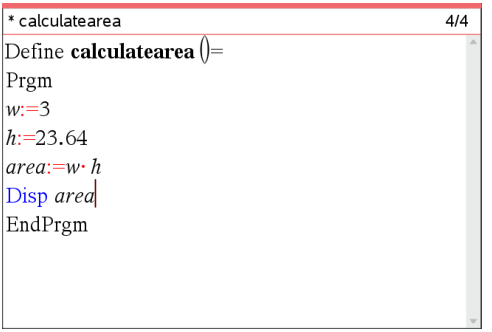
Hente værdier ind i et program

Du kan vælge blandt flere metoder til at give de værdier, der bruges af en funktions eller et programs beregninger.

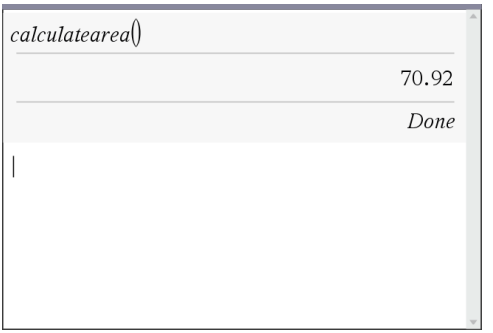
Indbygge værdierne i programmet eller funktionen

Denne metode er hovedsageligt nyttig til værdier, der skal være de samme, hver gang programmet eller funktionen anvendes.

1. Definer programmet.



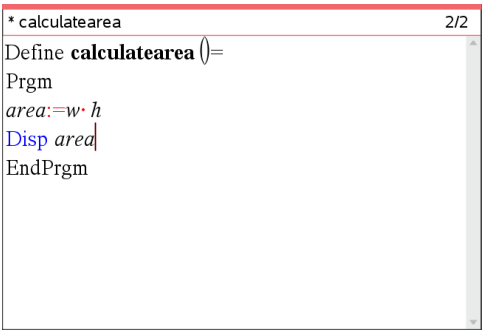
2. Kør programmet.



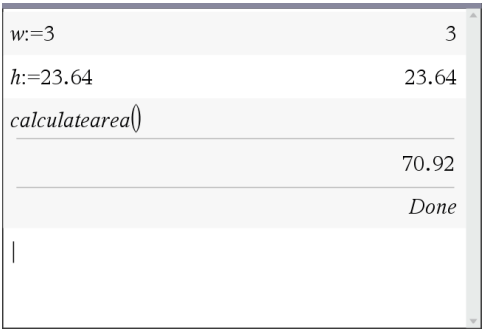
Lade brugeren tildele værdierne til variable

Programmer og funktioner kan referere til variable, der er oprettet i forvejen. Denne metode kræver, at brugerne husker variabelnavnene og tildeler dem værdier, før objektet anvendes.

1. Definer programmet.



2. Angiv variablene, og kør derefter programmet.

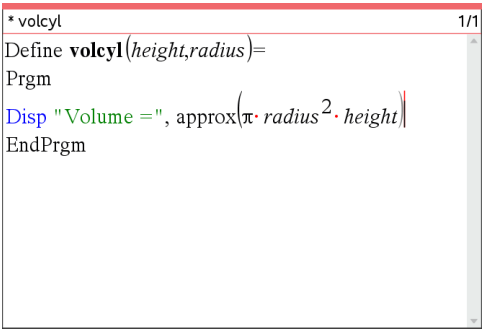


Lade brugeren angive værdierne som argumenter

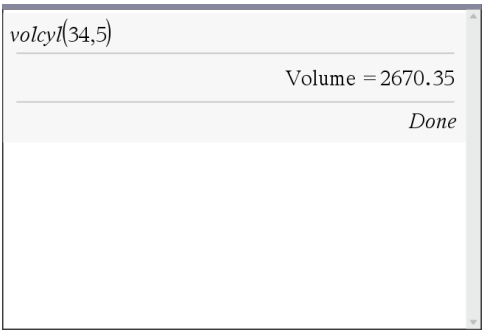
Med denne metode kan brugeren give en eller flere værdier som argumenter i udtrykket, der kalder programmet eller funktionen.

Følgende program, **volcyl**, beregner volumen for en cylinder. Det kræver, at brugeren angiver to værdier: højde og radius på cylinderen.

1. Definer **volcyl**-programmet.



2. Kør programmet, så det viser volumen på en cylinder med en højde på 34 mm og en radius på 5 mm.



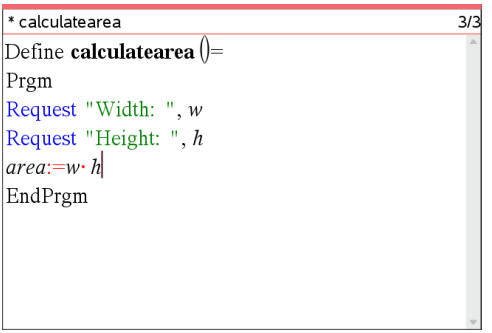
Bemærk: Du behøver ikke anvende parameternavnene, når du kører **volcyl**-programmet, men du skal give to argumenter (som værdier, variable eller udtryk). Den første skal repræsentere højden, og den anden skal repræsentere radius.

Anmodning af værdierne fra brugeren (kun programmer)

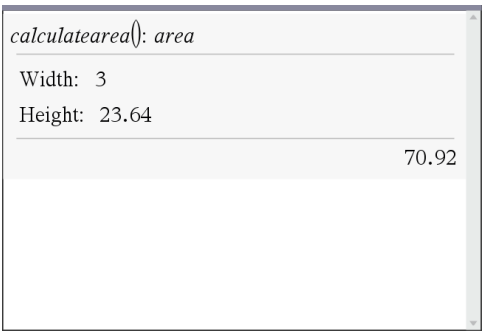
Med kommandoerne **Request** og **RequestStr** i et program kan du få programmet til at standse midlertidigt og vise en dialogboks, der beder brugeren om oplysninger. Denne metode kræver ikke, at brugeren husker variabelnavne eller den rækkefølge, de skal bruges i.

Kommandoerne **Request** og **RequestStr** kan ikke bruges i en funktion.

1. Definer programmet.



2. Kør programmet og svar på forespørgslerne.

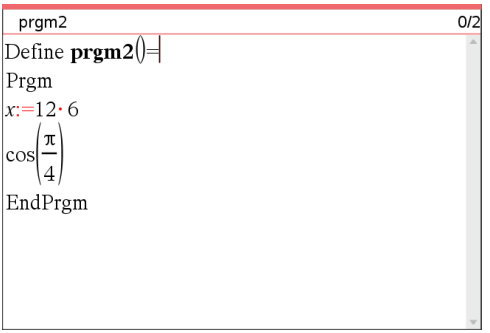


Brug **RequestStr** i stedet for **Request**, når du vil have programmet til at fortolke brugerens svar som en tegnstreng i stedet for et matematisk udtryk. Dermed undgås at bede brugeren om at omslutte svaret i citationstegn (““).

Visning af oplysninger

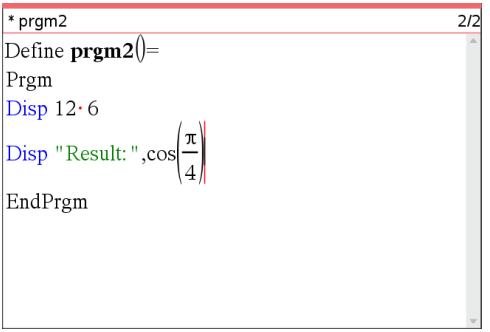
En funktion eller et program under eksekvering viser ikke beregnede mellemresultater, medmindre du medtager en kommando, der viser dem. Dette er en vigtig forskel mellem at udføre en beregning i indtastningslinjen og udføre den i en funktion eller et program.

Følgende beregning er viser for eksempel ikke et resultat i en funktion eller et program (selvom de gør det i indtastningslinjen).



Visning af oplysninger i historikken

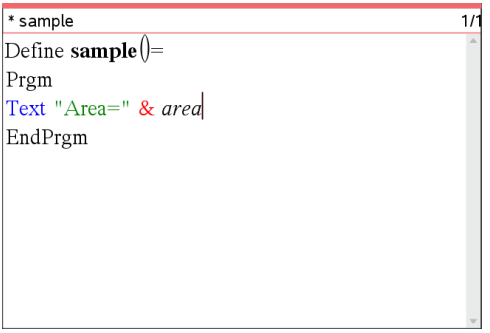
Med kommandoen **Disp** i et program eller en funktion kan du vise oplysninger som f.eks. mellemresultaterne i historikken.



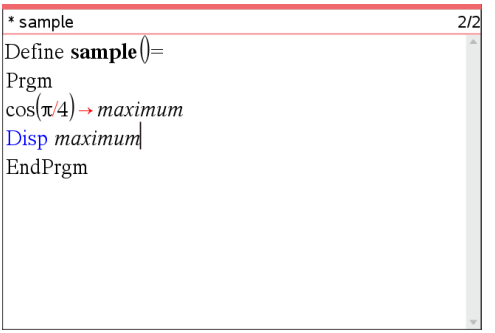
Visning af oplysninger i en dialogboks

Med kommandoen **Text** kan du standse et program midlertidigt og vise oplysninger i en dialogboks. Brugeren vælger **OK** for at fortsætte eller vælger **Annuller** for at standse programmet.

Kommandoen **Text** kan ikke bruges i en funktion.



Bemærk: Visning af et resultat med kommandoerne **Disp** eller **Text** lagrer ikke resultatet. Hvis du regner med at skulle kalde et resultat senere, skal du gemme det i en global variabel.

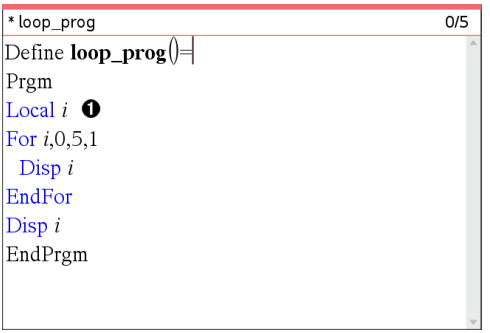


Anvendelse af lokale variable

En lokal variabel er en midlertidig variabel, der kun findes, mens en brugerdefineret funktion evalueres, eller der kører et brugerdefineret program.

Eksempel på en lokal variabel

Følgende programsegment viser en **For...EndFor-løkke** (der beskrives senere i dette modul). Variablen *i* er løkketælleren i løkken. I de fleste tilfælde anvendes variablen *i* kun, mens programmet kører.



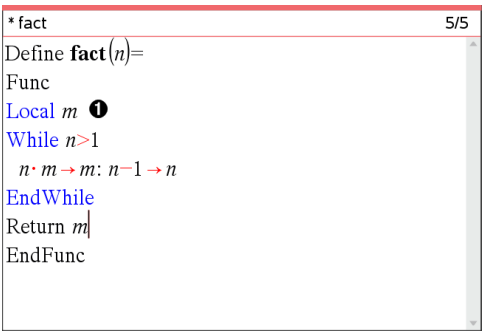
❶ Erklærer variablen i som lokal.

Bemærk: Når det er muligt, skal du erklære alle variable, der kun anvendes i programmet og ikke skal være tilgængelige, når programmet er stoppet, som lokale.

Hvad forårsager en fejlmeddelelse om en udefineret variabel?

En fejlmeddelelse om en **Udefineret** variabel vises, når du evaluerer en brugerdefineret funktion eller kører et brugerdefineret program, der kalder en lokal variabel, der ikke er initialiseret (tildelt en værdi).

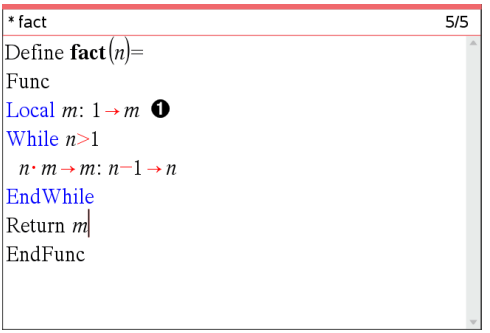
For eksempel:



❶ Den lokale variabel m er ikke tildelt en startværdi.

Initialiser lokale variable

Alle lokale variable skal tildeles en startværdi, før de kaldes.



❶ 1 gemmes som startværdi for m.

Bemærk (CAS): Funktioner og programmer kan ikke anvende en lokal variabel til at udføre symbolske beregninger.

CAS: Udføring af symbolske beregninger

Hvis en funktion eller et program skal udføre symbolske beregninger, skal du anvende en global variabel i stedet for en lokal. Du skal dog være sikker på, at den globale variabel ikke i forvejen findes uden for programmet. Følgende metoder kan hjælpe.

- Kalde en globalt variabelnavn, typisk med to eller flere tegn, der ikke risikerer at findes uden for funktionen eller programmet.
- Medtag **DelVar** i et program for at slette den globale variabel, hvis den findes, før den kaldes. (**DelVar** sletter ikke låste eller linkede variable.)

Forskelle mellem funktioner og programmer

En funktion, der er defineret i programeditoren, svarer meget til de funktioner, der er bygget ind i TI-Nspire™-softwaren.

- Funktioner skal returnere et resultat, der kan tegnes eller indsættes i en tabel. Programmer kan ikke returnere et resultat.
- Du kan anvende en funktion (men ikke et program) i et udtryk. For eksempel: **3 • func1(3)** er gyldig men ikke **3 • prog1(3)**.
- Du kan kun køre programmer fra applikationerne Regner og Noter. Du kan dog beregne funktioner i Regner, Noter, Lister og regneark, Grafer, Geometri og Data og statistik.
- En funktion kan kalde enhver variabel. Den kan dog kun lagre en værdi i en lokal variabel. Programs kan lagre i lokale og globale variable.

  Bemærk: Argumenter, der anvendes til at overføre værdier til en funktion, behandles automatisk som lokale variable. Hvis du vil lagre til andre variable, skal de erklæres som **Local** i funktionen.
- En funktion kan ikke kalde et program som en delrutine, men den kan kalde en anden brugerdefineret funktion.

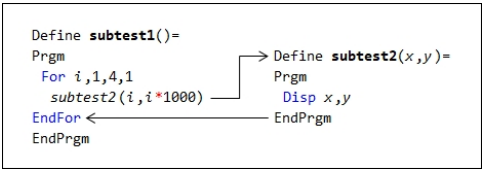
- Du kan ikke definere et program i en funktion.
- En funktion kan ikke definere en global funktion, men den kan definere en lokal funktion.

Kald mellem programmer

Programmer kan kalde hinanden som en subrutine. Subrutinen kan være ekstern (et separat program) eller intern (som en del af hovedprogrammet). Subrutiner er nyttige, når et program skal gentage den samme gruppe af kommandoer på flere forskellige steder.

Kald af et separat program

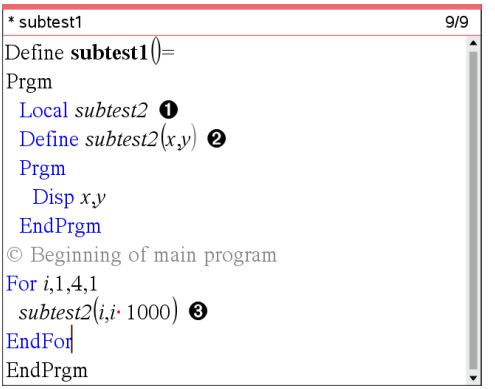
Du kan kalde et separat program, anvende samme syntaks, som når du plejer at køre programmet fra indtastningslinjen.



Definition og kald af en intern subrutine

Til at definere en intern subrutine skal du anvende kommandoen **Define** med **Prgm...EndPrgm**. Da en subrutine skal defineres, før den kan kaldes, er det god praksis at definere subrutiner i starten af hovedprogrammet.

En intern subrutine kaldes og udføres på samme måde som et separat program.



❶ Erklærer subrutinen som en lokal variabel.

❷ Definerer subrutinen.

❸ Kalder subrutinen.

Bemærk: Anvend programeditorens **Var**-menu til at indtaste kommandoerne **Define** og **Prgm...EndPrgm**.

Bemærkninger om anvendelse af subrutiner

Ved slutningen af en subrutine vender eksekveringen tilbage til det program, der kaldte den. Subrutinen kan afsluttes på ethvert andet tidspunkt ved at anvende **Return** uden noget argument.

En subrutine kan ikke åbne lokale variable, der er erklæret i det program, der kalder op. På samme måde kan det program, der kalder, ikke åbne lokale variable, der er erklæret i en subrutine.

Lbl-kommandoer er lokale for de programmer, de er placeret i. Derfor kan en **Goto**-kommando i det program, der kalder, ikke gå til et mærke i en subrutine eller omvendt.

Undgå fejl med cirkulær definition

Ved evaluering af en brugerdefineret funktion eller kørsel af et program kan du angive et argument, der omfatter samme variabel, der blev anvendt til at definere funktionen eller oprette programmet. Men for at undgå cirkulære definitionsfejl skal du tildele en værdi for variable, der anvendes i evaluering af funktionen eller kørslen af programmet. For eksempel:

$x+1 \rightarrow x$ ❶

– eller –

```
For i,i,10,1
  Disp i ❶
EndFor
```

❶ Udløser fejlmeddelelsen **Cirkulær definition (Circular definition)**, hvis x eller i ikke har en værdi. Fejlen optræder ikke, hvis x eller i i forvejen er tildelt en værdi.

Kontrol af flowet i en funktion eller et program

Når du kører et program eller evaluerer en funktion, eksekveres programlinjerne i rækkefølge. Visse kommandoer kan dog påvirke programafviklingen. For eksempel:

- Kontrolkonstruktioner som **If...EndIf**-kommandoer benytter en betingelsestest til at bestemme, hvilken del af et program, der skal eksekveres.
- Løkkekommandoer som **For...EndFor** gentager en gruppe kommandoer.

Anvendelse af If, Lbl og Goto til at styre programafviklingen

Med kommandoen **If** og en række **If...EndIf**-konstruktioner kan du udføre en sætning eller blok af sætninger betinget, det vil sige, baseret på resultatet af en test (som f.eks. x>5). **Med Lbl** (Mærke) og **Goto**-kommandoerne kan du gå til eller springe mellem stederne i en funktion eller et program.

Kommandoen **If** og en række **If...EndIf**-konstruktioner findes i programeditorens **Control**-menu.

Når du indsætter en konstruktion som **If...Then...EndIf**, indsættes en skabelon på markørstedet. Markøren er placeret, så du kan indsætte en betingelsestest.

If- kommando

Til udførelse af en enkelt kommando, når en betingelsestest er sand, anvendes den generelle form:

```
If x>5
  Disp "x is greater than 5" ❶
Disp x ❷
```

❶ Udføres kun, hvis x>5. Ellers springes sætningen over

❷ Viser altid værdien af x.

I dette eksempel skal du lagre en værdi i x, før **If** -kommandoen udføres.

If...Then...EndIf-konstruktioner

Til udførelse af en gruppe kommandoer, når en betingelsestest er sand, anvendes konstruktionen:

```
If x>5 Then
  Disp "x is greater than 5" ❶
  2·x→x ❶
EndIf
Disp x ❷
```

❶ Udføres kun, hvis x>5.

❷ Viser værdien af:
2x hvis x>5
x hvis x≤5

Bemærk: EndIf markerer slutningen på **Then**-blok, der udføres, hvis betingelsen er sand.

If...Then...Else... EndIf-konstruktioner

Anvend denne konstruktion, hvis der skal udføres en gruppe kommandoer, hvis en betingelsestest er sand, og en anden gruppe, hvis betingelsen er falsk:

```
If x>5 Then
  Disp "x is greater than 5 " ❶
  2·x→x ❶
Else
  Disp "x is less than or equal to 5 " ❷
  5·x→x ❷
EndIf
Disp x ❸
```

❶ Udføres kun, hvis x>5..

❷ Udføres kun, hvis x≤5.

❸ Viser værdien på:
2x hvis x>5
5x hvis x≤5

If...Then...ElseIf... EndIf-konstruktioner

Med en mere kompleks form af If-kommandoen kan du teste for flere betingelser. Hvis du ønsker et program til at teste et argument leveret af brugeren, der finder et af fire valg.

For at teste for hvert valg (If Choice=1, If Choice=2 og så videre) anvendes **If...Then...ElseIf...EndIf**-konstruktionen.

Lbl og Goto-kommandoer

Du kan også styre afviklingen ved hjælp af **Lbl** (Mærke) og **Goto**-kommandoerne. Disse kommandoer findes i programeditorens menu **Transfers**.

Med **Lbl**-kommandoen kan du mærke (give et navn til) en bestemt placering i funktionen eller programmet.

Lbl *labelName*	Navn, der gives til dette sted (anvend samme navngivningskonventioner som et variabelnavn)

Du kan derefter anvende **Goto**-kommandoen ethvert sted i funktionen eller programmet til at gå til det sted, der svarer til det angivne mærke.

Goto *labelName*	Angiver, hvilken **Lbl**-kommando, der skal gås til

Da **Goto**-kommandoen er ubetinget (den går altid til det angivne mærke), anvendes den ofte med en **If**-kommando, så du kan angive en betingelsestest. For eksempel:

```
If x>5
  Goto GT5 ❶
Disp x

.............

Lbl GT5 ❷
Disp "The number was > 5"
```

❶ Hvis x>5, gås direkte til mærket GT5.

❷ I dette eksempel skal programmet indeholde kommandoer (som **Stop**), der forhindrer **Lbl** GT5 i at blive udført, hvis x≤5.

Anvendelse af løkker til at gentage en gruppe kommandoer

Til at angive samme gruppe kommandoer i en rækkefølge anvendes en af løkkekonstruktionerne. Der kan anvendes flere typer af løkker. Hver type giver dig en ny måde på at bryde løkken ud fra en betingelsestest.

Løkker og løkkerelaterede kommandoer findes i programeditorens menuer **Control** og **Transfers**.

Når du indsætter en af løkkekonstruktionerne, indsættes dens skabelon på markørens position. Du kan så begynde at indsætte de kommandoer, der udføres i løkken.

For...EndFor-løkker

En **For...EndFor**-løkke benytter en tæller til at styre det antal gange, løkken skal gentages. Syntaksen på **For**-kommandoen er:

Bemærk: Slutværdien kan være mindre end startværdien, hvis trinfølgen er negativ.

For *variabel, start, slut* [, *trinstørrelse*]

❶ ❷ ❸ ❹

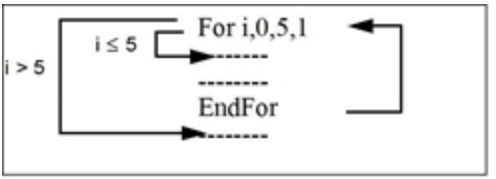
❶ *Variabel*, der anvendes som en tæller

❷ Tællerværdi, der anvendes første gang, **For** eksekveres

❸ Afslutter løkken, når *variabel* overskrider denne værdi

❹ Føjes til tælleren for hver gang, **For** eksekveres (Hvis denne valgfrie værdi udelades, sættes *trinstørrelsen* til 1.)

Når **For** udføres, sammenlignes *variabel* med *end*-værdien. Hvis *variabel* ikke overskrider *end*, udføres løkken. Ellers, springer styringen til kommandoen, der følger efter **EndFor**.



Bemærk: For-kommandoen øger automatisk tællervariablen trinvist, så funktionen eller programmet kan afslutte løkken efter et bestemt antal gentagelser.

Ved slutningen af løkken (**EndFor**), springer kontollen tilbage til **For**-kommandoen, hvor variablen øges og sammenlignes med *end*.

For eksempel:

```
For i,0,5,1
  Disp i ❶
EndFor
Disp i ❷
```

❶ Viser 0, 1, 2, 3, 4 og 5.

❷ Viser 6. Når *variabel* er øget trinvist til 6 udføres løkken ikke mere.

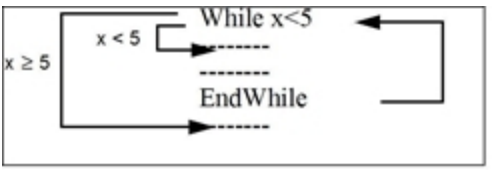
Bemærk: Du kan erklære tællervariablen som lokal, hvis den ikke skal gemmes, når funktionen eller programmet standser.

While...EndWhile-løkker

While...EndWhile-løkker gentager en kommandoblok, så længe en angivet betingelse er sand. Syntaksen på **While**-kommandoen er:

While -*betingelse*

Når **While** udføres, evalueres *betingelse*. Hvis *betingelse* er sand, udføres løkken. Ellers springer kontrollen til kommandoen efter **EndWhile**.



Bemærk: While-kommandoen ændrer ikke automatisk betingelsen. Du skal medtage kommandoer, der lader funktionen eller programmet afslutte løkken.

Ved slutningen af løkken (**EndWhile**), springer kontrollen tilbage til **While**-kommandoen, hvor betingelsen evalueres igen.

For løkken skal køre første gang skal betingelsen i starten være sand.

- Alle kaldte variable, der kaldes i betingelsen, skal være indstillet før **While**-kommandoen. (Du kan indbygge værdierne i funktionen eller programmet, eller du kan bede brugeren om at indsætte værdierne.)
- Løkken skal indeholde kommandoer, der ændrer værdierne i betingelsen, så den tilsidst bliver falsk. Ellers er betingelsen altid sand, og funktion eller programmet kan ikke afslutte løkken (en såkaldt uendelig løkke).

For eksempel:

```
0→x ❶
While x<5
  Disp x ❷
  x+1→x ❸
EndWhile
Disp x ❹
```

❶ Indstiller først x.

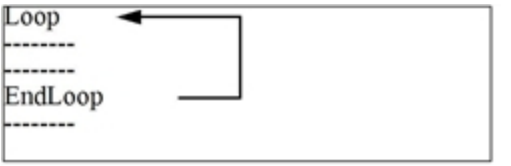
❷ Viser 0, 1, 2, 3 og 4.

❸ Øger x trinvist.

❹ Viser 5. Når x øges til 5, udføres løkken ikke.

Loop...EndLoop-løkker

En **Loop...EndLoop** giver en uendelig løkkem der gentages uendeligt. **Loop**-kommandoen bruger ikke argumenter.



Man indsætter kommandoer i den løkke, hvormed programmet afslutter løkken. Almindeligt anvendte kommandoer er: **If**, **Exit**, **Goto** og **Lbl** (mærke). For eksempel:

❶ En **If**-kommando styrer betingelsen.

❷ Afslutter løkken og springer hertil, når x er steget til 6.

Bemærk: Exit-kommandoen afslutter den aktuelle løkke.

I dette eksempel kan **If** kommandoen være ethvert sted i løkken.

Når If-kommandoen er:	Er løkken:
I starten af løkken	Udføres kun, hvis betingelsen er sand.
I slutningen af løkken	Udføres mindst en gang og gentages kun, hvis betingelsen er sand.

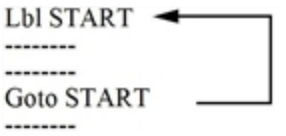
If-kommandoen kunne også brugen en **Goto** kommando til at overføre programstyringen til en angivet **Lbl** (etiket)-kommando.

Gentager løkken straks

Cycle-kommandoen overfører straks programstyringen til næste gentagelse af en løkke (før den aktuelle løkke er færdig). Denne kommando fungerer med **For...EndFor**, **While...EndWhile** og **Loop...EndLoop**.

Lbl og Goto-løkker

Selvom **Lbl** (mærke) og **Goto**-kommandoerne egentlig ikke blot er løkke, kan de anvendes til at lave en uendelig løkke. For eksempel:



Som med **Loop...EndLoop**, skal løkken indeholder kommandoer til at lade funktionen eller programmet afslutte løkken.

Ændring af tilstandsindstillinger

Funktioner og programmer kan benytte **setMode()**-funktionen til midlertidigt at indstille bestemte beregnings- eller resultattilstande. Menuen **Tilstand (Mode)** i programeditoren gør det nemt at indsætte den korrekte syntaks, uden at du skal huske talkoder.

Bemærk: Tilstandsskift, der er foretaget i en funktion eller programdefinition, fortsætter uden for funktionen eller programmet.

Indstilling af en tilstand

1. Placer markøren, hvor du vil indsætte funktionen **setMode**.
2. I menuen **Tilstand** kan du vælge den tilstand, der skal ændres, og vælg den nye indstilling.

   Den korrekte syntaks indsættes ved markørens placering. For eksempel:

```
setMode(1,3)
```

Fejlfinding af programmer og håndteringsfejl

Når du har skrevet en funktion eller et program, kan du benytte en række teknikker til at søge og rette fejl. Du kan også indbygge en fejlhåndteringskommando i selve funktionen eller programmet.

Hvis funktionen eller programmet lader brugeren vælge blandt flere indstillinger, skal du køre det og teste hver indstilling.

Teknikker til fejlsøgning

Fejlmeddelelser under kørslen kan finde syntaksfejl men ikke fejl i programlogikken. Følgende teknikker kan være nyttige.

- Indsæt midlertidigt **Disp**-kommandoer til at vise værdierne af kritiske variable.
- Du kan kontrollere, at en løkke udføres det korrekte antal gange ved at anvende **Disp** til at vise tællervariablen eller værdier i betingelsestesten.
- Du kan bekræfte, at en subrutine udføres, ved at anvende **Disp** til at vise meddelelser som "Starter subrutine" og "Afslutter subrutine" ved start og slut på subrutinen.
- For at stoppe et program eller en funktion manuelt,
  - Windows®: Hold tasten **F12** nede, mens der gentagne gange trykkes på **Enter**.
  - Macintosh®: Hold tasten **F5** nede, mens der gentagne gange trykkes på **Enter**.
  - Håndholdt: Hold tasten [on] nede, mens der gentagne gange trykkes på [enter].

Fejlhåndteringskommandoer

Kommando	Beskrivelse
Try...EndTry	Definerer en blok, der lader en funktion eller et program udføre en kommando og om nødvendigt reparere efter en fejl, der er genereret fra en fejl genereret med denne kommando.
ClrErr	Sletter fejlstatus og sætter systemvariabelen *errCode til nul. Vi henviser til punktet om kommandoen Try i* ***Opslagsvejledningen for et eksempel på brugen af*** *errCode.*
PassErr	Overfører en fejl til næste niveau i **Try...EndTry**-blokken.

Generelle oplysninger

Online hjælp

education.ti.com/eguide

Vælg dit land for at få flere produktoplysninger.

Kontakt TI-Support

education.ti.com/ti-cares

Vælg dit land for at finde ressourcer for teknisk support og andre supportressourcer.

Service og garanti

education.ti.com/warranty

Vælg dit land for at få oplysninger om varigheden og betingelserne for garantien, eller om produktservice.

Begrænset reklamationsret. Denne garanti påvirker ikke dine lovbestemte rettigheder.

Texas Instruments Incorporated

12500 TI Blvd.

Dallas, TX 75243