TI-Nspire™ programredaktör
Bruksbok

Mer information om TI-teknologi hittar du i online-hjälpen på education.ti.com/eguide.

Viktig information

Med undantag för vad som uttryckligen anges i licensen som följer med ett program, ger Texas Instruments ingen garanti, varken uttrycklig eller underförstådd, inklusive men inte begränsat till några underförstådda garantier för säljbarhet och lämplighet för ett visst ändamål, avseende program eller bokmaterial och gör sådana material som är tillgängligt endast på "as-is" -basis. Under inga omständigheter ska Texas Instruments vara ansvarig för någon för speciella, säkerheter, tillfälliga eller följdskador i samband med eller som uppstår genom inköp eller användning av dessa material och Texas Instruments ensamrätt, oavsett form av åtgärd, får inte överstiga det belopp som anges i licensen för programmet. Dessutom ska Texas Instruments inte vara ansvarig för något påstående av något slag mot användningen av dessa material av någon annan part.

TI-Nspire™ -programvaran använder Lua som skriptmiljö. För information om upphovsrätt och licens, se http://www.lua.org/license.html.

TI-Nspire™ -programvaran använder Chipmunk Physics version 5.3.4 som simuleringsmiljö. För licensinformation, se http://chipmunk-physics.net/release/Chipmunk-5.x/Chipmunk-5.3.4-Docs/.

Microsoft® och Windows® är registrerade varumärken som tillhör Microsoft Corporation i USA och / eller andra länder.

Mac OS®, iPad® och OS X® är registrerade varumärken som tillhör Apple Inc.

Unicode® är ett registrerat varumärke tillhörande Unicode, Inc. i USA och andra länder.

De faktiska produkterna kan variera något från de visade bilderna.

Innehåll

Komma igång med Programeditorn

Du kan skapa användardefinierade funktioner och program genom att skriva in definitionspåståenden i inmatningsraden i applikationen Räknare eller genom att använda Programeditorn. Programeditorn erbjuder vissa fördelar som beskrivs i detta avsnitt. Se *Räknare* för mer information.

- Editorn har programmeringsmallar och dialogrutor för att hjälpa dig definiera funktioner och program med rätt syntax.
- Editorn låter dig mata in flerradiga programsatser utan krav på någon speciell knappsekvens för att lägga till varje rad.
- Du kan enkelt skapa privata och allmänna biblioteksobjekt (variabler, funktioner och program). Se *Bibliotek* för mer information.

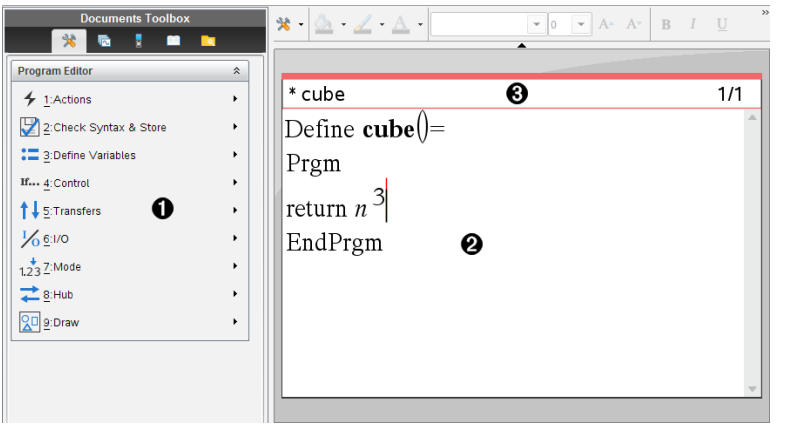
Öppna Programeditorn

▶ Så här lägger du till en Programeditor-sida i det nuvarande problemet:

I verktygsfältet klickar du på **Infoga > Programeditorn > Ny**.

Handenhet: Tryck på doc▾ och välj **Infoga > Programeditor > Ny**.

Obs: Man kan också komma åt editorn från menyn **Funktioner & Program** på en Räknare-sida.



❶ Menyn Programeditor – Denna meny är alltid tillgänglig när du är i Programeditorns arbetsområde i läget Normal vy.

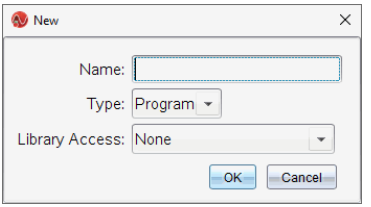
❷ Programeditorns arbetsyta

❸ Statusrad som visar information om radnummer och namnet på funktionen eller programmet som redigeras. En asterisk (*) indikerar att denna funktion är ”smutsig”, vilket innebär att den har ändrats sedan dess syntax kontrollerades senaste gången och den har lagrats.

Definiera ett program eller en funktion

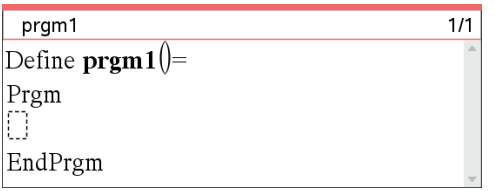
Starta en ny Programeditor

1. Säkerställ att du är i dokumentet och problemet där du vill skapa programmet eller funktionen.
2. Klicka på knappen **Infoga** på applikationens verktygsfält och välj **Programeditor > Ny**. På handenheten trycker du på doc och väljer **Infoga > Programeditor > Ny**.)



3. Skriv in ett namn på funktionen eller programmet som du definierar.
4. Välj **Typ** (**Program** eller **Funktion**).
5. Ställ in **Biblioteksåtkomst**:
   - Välj **None** om du vill använda funktionen eller programmet endast från det aktuella dokumentet och problemet.
   - Tryck på **LibPriv** om du vill att funktionen eller programmet skall kunna nås från alla dokument, men inte visas i Katalogen.
   - Tryck på **LibPriv (Visa i Katalog)** om du vill att funktionen eller programmet skall kunna nås från alla dokument och visas i Katalogen. Se *Bibliotek* för mer information.
6. Klicka på **OK**.

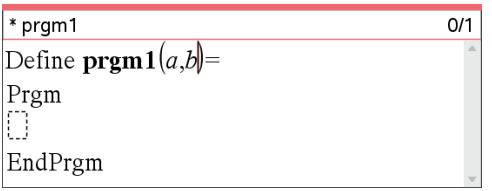
   Programeditorn öppnas i ett nytt fönster med en mall som matchar de val som du har gjort.



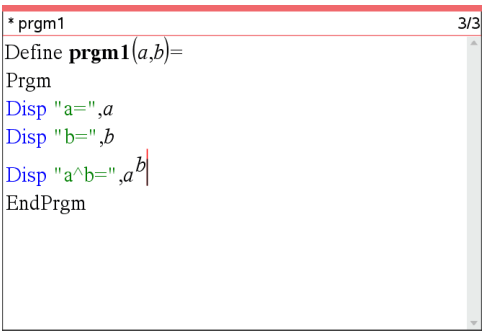
Infoga rader i en funktion eller ett program

Programeditorn utför inte kommandon eller utvärderar uttryck när du skriver in dem. De utförs endast när du utvärderar funktionen eller kör programmet.

1. Om din funktion eller ditt program kräver att användaren anger argument, skriv in parameternamnen i parentesen som kommer efter namnet. Separera parametrarna med kommatecken.



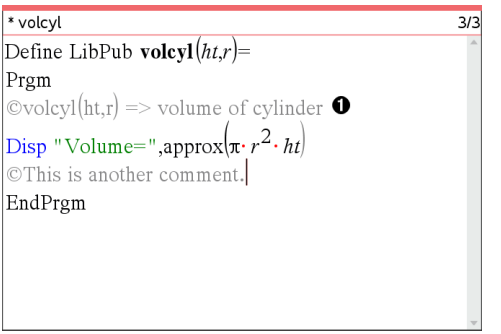
2. Mellan raderna Func och EndFunc (eller Prgm och EndPrgm), för in dina rader med satser som bildar din funktion eller ditt program.



- Du kan antingen skriva in namnen på funktioner och kommandon eller infoga dem från Katalogen.
- Om en rad är längre än skärmens bredd kan du skrolla för att se hela uttrycket.
- Tryck på **Enter** när varje rad är klar. Detta infogar en ny, tom rad så att du kan fortsätta skriva in ytterligare en programrad.
- Använd piltangenterna ◄, ►, ▲, och ▼ för att skrolla genom funktionen eller programmet och infoga eller redigera kommandon.

Infoga kommentarer

Kommentarer kan vara användbara för någon som visar eller redigerar programmet. De visas inte när programmet körs och de påverkar inte programflödet. Symbolen © visas i början av kommentarsraden.



❶ Kommentar som visar den syntax som krävs. Eftersom detta biblioteksobjekt är allmänt och denna kommentar är på den första raden i ett Func- eller Prgm-block visas kommentaren i Katalogen som hjälp. Se *Bibliotek* för mer information.

För att infoga en kommentar:

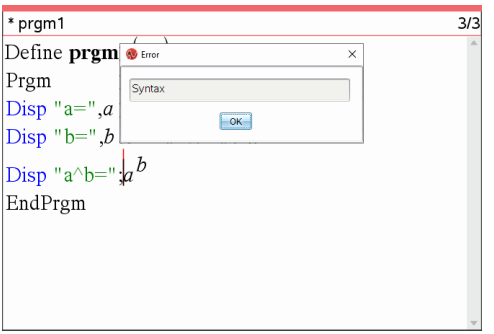
1. Placera markören i slutet på raden där du vill infoga en kommentar.
2. Från menyn **Åtgärder**, klicka på **Infoga kommentar**, eller tryck **Ctrl+T**.
3. Skriv in kommentaren efter ©-symbolen.

Kontrollera syntax

Programeditorn kontrollerar om funktionen eller programmet har korrekt syntax.

- På menyn **Kontrollera syntax och lagra**, klicka på **Kontrollera syntax**.

  Om syntaxkontrollen hittar syntaxfel visar den ett felmeddelande och försöker placera markören nära det första felet så att du kan korrigera det.



Lagra funktionen eller programmet

Du måste lagra din funktion eller ditt program för att göra den/det åtkomlig(t). Programeditorn kontrollerar automatiskt syntaxen innan lagring.

En asterisk (*) visas längst upp till vänster i Programredigeraren för att markera att funktionen eller programmet inte har lagrats.

- På menyn **Kontrollera syntax och lagra**, klicka på **Kontrollera syntax och lagra**.

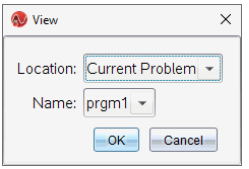
  Om syntaxkontrollen hittar syntaxfel visar den ett felmeddelande och försöker placera markören nära det första felet.

  Om inga syntaxfel hittas visas meddelandet "programnamn lagring lyckades" på statusraden längst upp i Programeditorn.

Obs: Om funktionen eller programmet definieras som ett biblioteksobjekt måste du också spara dokumentet i den utpekade biblioteksmappen och uppdatera biblioteken för att göra objektet åtkomligt för andra dokument. Se *Bibliotek* för mer information.

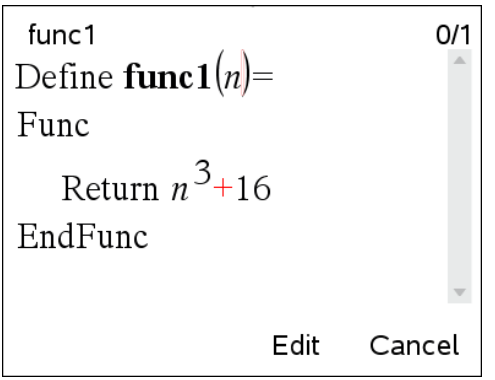
Visa ett program eller en funktion

1. På menyn **Åtgärder**, välj **Visa**.



2. Om funktionen eller programmet är ett biblioteksobjekt, välj dess bibliotek på listan **Location**.
3. Välj funktionens eller programmets namn på **Name**-listan.

   Funktionen eller programmet visas i en granskare.



4. Använd piltangenterna för att visa funktionen eller programmet.
5. Om du vill redigera programmet, klicka på **Redigera**.

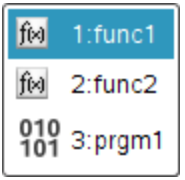
   Obs: Alternativet **Edit** är endast tillgängligt för funktioner och program som är definierade i det aktuella problemet. För att redigera ett biblioteksobjekt måste du först öppna dess biblioteksdokument.

Öppna en funktion eller ett program för redigering

Du kan endast öppna en funktion eller ett program från det aktuella problemet.

Obs: Du kan inte modifiera ett låst program eller en låst funktion. För att låsa upp objektet, gå till en Räknare-sida och använd kommandot **Lås upp**.

1. Visa listan på tillgängliga funktioner och program.
   - På menyn **Åtgärder**, välj **Öppna**.

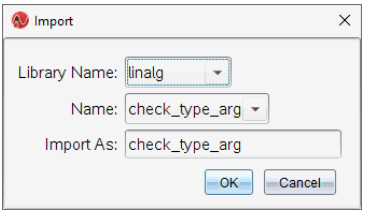


2. Välj objektet som skall öppnas.

Importera ett program från ett bibliotek

Du kan importera en funktion eller ett program som definierats som biblioteksobjekt till Programeditorn inom det aktuella problemet. Den importerade kopian är inte låst även om originalet är låst.

1. På menyn **Åtgärder**, välj **Importera**.



2. Välj **Library Name**.
3. Välj objektets **Name**.
4. Om du vill att det importerade objektet skall ha ett annat namn, skriv in namnet under **Import As**.

Skapa en kopia av en funktion eller ett program

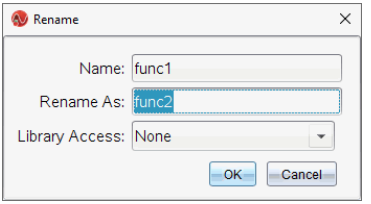
När du skapar en ny funktion eller nytt program kan det vara enklare att starta med en kopia av den/det aktuella funktionen/problemet. Kopian som du skapar är inte låst även om originalet är låst.

1. På menyn **Åtgärder**, välj **Skapa kopia**.
2. Skriv in ett nytt namn eller klicka på **OK** för att godkänna det föreslagna namnet.
3. Om du vill ändra åtkomstnivån, välj **Library Access** och välj sedan en ny nivå.

Ändra namn på ett program eller en funktion

Du kan ändra namn och (alternativt) åtkomstnivå för den aktuella funktionen eller programmet.

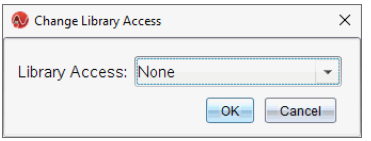
1. På menyn **Åtgärder**, välj **Byt namn**.



2. Skriv in ett nytt namn eller klicka på **OK** för att godkänna det föreslagna namnet.
3. Om du vill ändra åtkomstnivån, välj **Library Access** och välj sedan en ny nivå.

Ändra åtkomstnivå för bibliotek

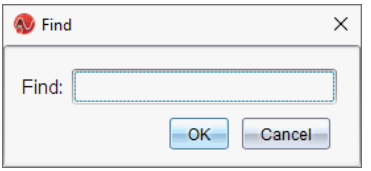
1. På menyn **Åtgärder**, välj **Ändra Biblioteksåtkomst**.



2. Välj **Library Access**:
   - Välj **None** om du vill använda funktionen eller programmet endast från det aktuella Calculator-problemet.
   - Välj **LibPriv** om du vill att funktionen eller programmet skall kunna nås från alla dokument, men inte visas i Katalogen.
   - Välj **LibPub** om du vill att funktionen eller programmet skall kunna nås från alla dokument och även visas i Katalogen.

Hitta text

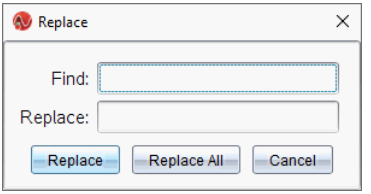
1. På menyn **Åtgärder**, välj **Sök**.



2. Skriv in texten du vill hitta och klicka på **OK**.
   - Om texten hittas markeras den i programmet.
   - Om texten inte hittas visas ett meddelande om detta.

Söka och ersätta text

1. På menyn **Åtgärder**, välj **Sök och ersätt**.



2. Skriv in texten du vill hitta.
3. Skriv in önskad text.
4. Klicka på **Replace** för att ersätta den första förekomsten efter markörens position eller klicka på **Replace all** för att ersätta alla förekomster.

   Obs: Om texten hittas i en matematikmall visas ett meddelande som varnar dig om att din ersättningstext kommer att ersätta hela mallen och inte bara den hittade texten.

Stänga den aktuella funktionen eller programmet

▸ På menyn **Åtgärder**, välj **Stäng**.

Om funktionen eller programmet har ändringar som ej lagrats uppmanas du att kontrollera syntaxen och lagra före stängning.

Köra program och utvärdera funktioner

Efter att ha definierat och lagrat ett program eller en funktion kan du använda det från en applikation. Alla applikationerna kan utvärdera funktioner men endast applikationerna Räknare och Anteckningar kan köra program.

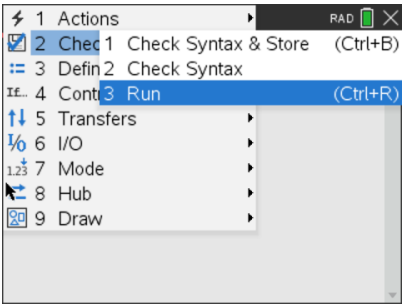
Programsatserna exekveras i sekventiell följd (vissa kommandon ändrar dock programflödet). Eventuellt resultat visas på applikationens arbetsyta.

- Programexekvering fortsätter tills den når den sista satsen eller ett **Stop**-kommando
- Funktionsexekvering fortsätter tills den når ett **Return**-kommando

Köra ett program eller en funktion från Programeditorn

1. Se till att du har definierat ett program eller en funktion och att Programeditorn är det aktiva fönstret (dator) eller den aktiva sidan (handenhet).
2. Klicka på knappen **Dokumentverktyg** på verktygsfältet  och välj **Kontrollera syntax och lagra > Kör**.
   —eller—
   Tryck på **Ctrl+R**.

   Handenhet: Tryck på menu 2 3, eller tryck på ctrl R.



Detta kommer automatiskt att:

- kontrollera syntaxen och lagra programmet eller funktionen,
- kopiera programmets eller funktionens namn till den första tillgänglig raden i applikationen Räknare direkt efter Programeditorn. Om ingen Räknare finns på den platsen infogas en ny.

3. Skriv in värdena eller variabelnamnen i parentesen om programmet eller funktionen kräver att du anger ett eller flera argument.
4. Tryck på enter.

Obs: Du kan också köra ett program eller en funktion i applikationerna Räknare eller Anteckningar genom att ange namnet på programmet med parenteser och eventuella argument som krävs samt trycka enter.

Använda korta och långa namn

Närhelst du är i samma problem där ett objekt är definierat kan du nå det genom att ange dess korta namn (namnet som givits i objektets kommande **Define**). Detta gäller för alla definierade objekt, inklusive privata, offentliga och icke-biblioteksobjekt.

Du kan nå ett biblioteksobjekt från valfritt dokument genom att ange objektets långa namn. Ett långt namn består av objektets biblioteksdokument följt av ett omvänt snedstreck "\" och sedan objektets namn. Till exempel är det långa namnet för objektet definierat som **func1** i biblioteksdokumentet **lib1 lib1\func1**. För att skriva tecknet "\" på handenheten trycker man shift ÷.

Obs: Om du inte kommer ihåg det exakta namnet, eller argumentordningen som krävs för ett privat biblioteksobjekt, kan du öppna biblioteksdokumentet eller använda Programeditorn för att visa objektet. Du kan också använda **getVarInfo** för att visa en lista över objekt i ett bibliotek.

Använda ett allmänt bibliotheksprogram eller en -funktion

1. Kontrollera att du har definierat objektet i dokumentets första problem, lagrat objektet, sparat biblioteksdokumentet i mappen MyLib och uppdaterat biblioteken.
2. Öppna den TI-Nspire™-applikation i vilken du vill använda funktionen eller programmet.

   Obs: Alla applikationer kan utvärdera funktioner, men endast applikationerna Räknare och Anteckningar kan köra program.

3. Öppna Katalogen och använd biblioteksfliken för att hitta och infoga objektet.
   —eller—
   Ange namnet för objektet. Avsluta alltid namnet med parenteser när det handlar oml en funktion eller ett program.

   ```
   libs2\func1()
   ```

4. Skriv in värdena eller variabelnamnen i parentesen om programmet eller funktionen kräver att du anger ett eller flera argument.

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

5. Tryck på enter.

Använda en privat biblioteksfunktion eller ett privat bibliotheksprogram

För att använda ett privat biblioteksobjekt måste du veta dess långa namn. Det långa namnet på exempelvis ett objekt definierat som **func1** i biblioteksdokumentet **lib1** är **lib1\func1**.

Obs: Om du inte kommer ihåg det exakta namnet eller argumentordningen som krävs för ett privat biblioteksobjekt kan du öppna biblioteksdokumentet eller använda Programeditorn för att visa objektet.

1. Kontrollera att du har definierat objektet i dokumentets första problem, lagrat objektet, sparat biblioteksdokumentet i mappen MyLib och uppdaterat biblioteken.
2. Öppna den TI-Nspire™-applikation i vilken du vill använda funktionen eller programmet.

   Obs: Alla applikationer kan utvärdera funktioner, men endast applikationerna Räknare och Anteckningar kan köra program.

3. Ange namnet för objektet. Avsluta alltid namnet med parenteser när det handlar om en funktion eller ett program.

   ```
   libs2\func1()
   ```

4. Skriv in värdena eller variabelnamnen i parentesen om funktionen eller programmet kräver att du anger ett eller flera argument.

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

5. Tryck på enter.

Avbryta ett pågående program eller en funktion

Medan ett program eller en funktion exekveras visas upptagetsymbolen ⏲.

- För att stoppa programmet eller funktionen,
  - Windows®: Håll ned **F12** och tryck på **Enter** upprepade gånger.
  - Mac®: Håll ned **F5** och tryck på **Enter** upprepade gånger.
  - Handenhet: Håll ned on och tryck på enter upprepade gånger.

  Ett meddelande visas. Välj **Gå till** för att redigera programmet eller funktionen i Programeditorn. Markören dyker upp vid kommandot där avbrottet uppstod.

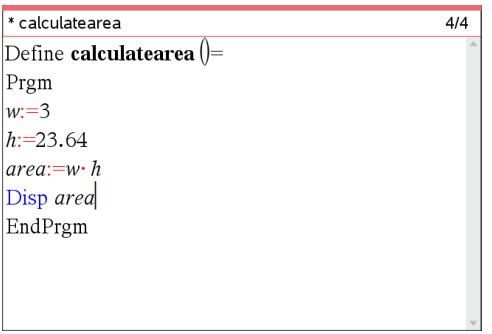
Mata in värden i ett program

Du kan välja mellan flera metoder för att mata in de värden som en funktion eller ett program använder i beräkningar.

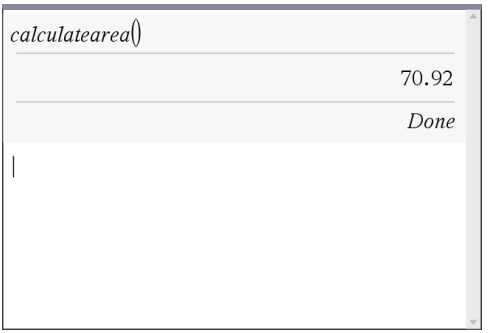
Bädda in värden i programmet eller funktionen

Denna metod är i första hand användbar för värden som måste vara samma varje gång programmet eller funktionen används.

1. Definiera programmet.



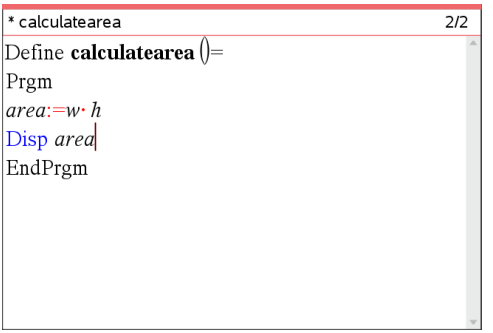
2. Kör programmet.



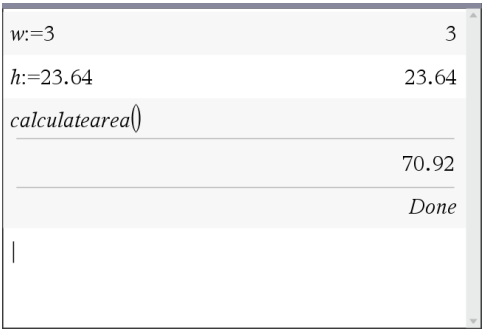
Låta användaren tilldela värden till variabler

Ett program eller en funktion kan referera till variabler som har skapats i förväg. Denna metod kräver att användaren kommer ihåg variabelnamnen och tilldelar värden till variablerna innan objektet används.

1. Definiera programmet.



2. Mata in variablerna och kör sedan programmet.

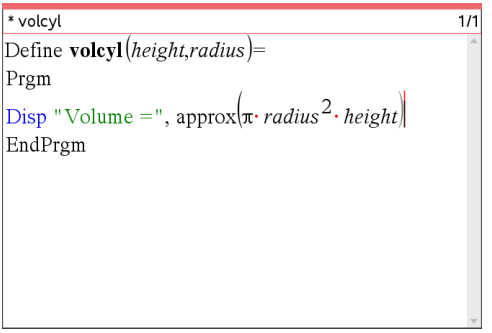


Låta användaren mata in värden som argument

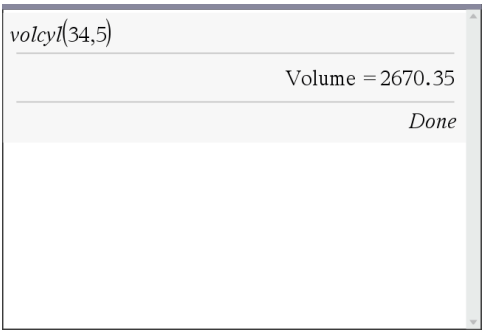
Denna metod låter användaren överföra ett eller flera värden som argument inom det uttryck som anropar programmet eller funktionen.

Följande program, **volcyl**, beräknar volymen hos en cylinder. Det kräver att användaren matar in två värden: cylinderns höjd och radie.

1. Definiera programmet **volcyl**.



2. Kör programmet för att visa volymen hos en cylinder med en höjd på 34 mm och en radie på 5 mm.



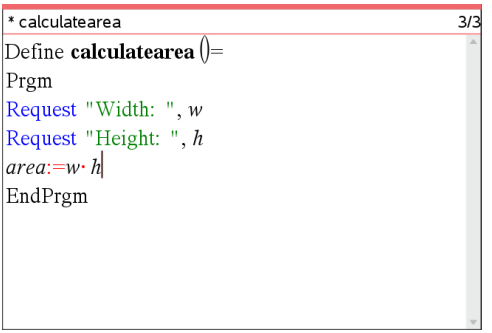
Obs: Du behöver inte använda parameternamnen när du kör programmet **volcyl**, men du måste ange två argument (som värden, variabler eller uttryck). Det första värdet måste representera höjden och det andra måste representera radien.

Begära värden från användaren (endast program)

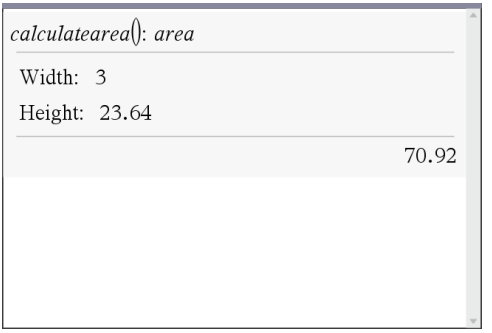
Du kan använda kommandona **Request** och **RequestStr** i ett program för att få programmet att göra en paus och visa en dialogruta som begär information från användaren. Denna metod kräver inte att användaren kommer ihåg variabelnamnen eller ordningen i vilken de behövs.

Du kan inte använda kommandot **Request** eller **RequestStr** i en funktion.

1. Definiera programmet.



2. Kör programmet och mata in de begärda uppgifterna.

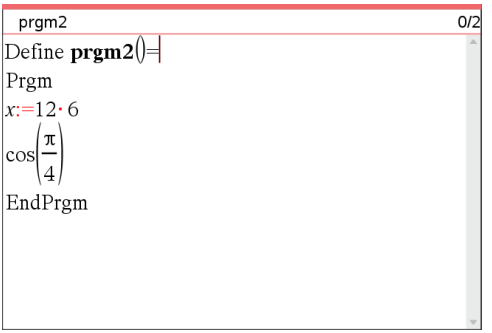


Använd **RequestStr** i stället för **Request** när du vill att programmet ska tolka användarens svar som en teckensträng i stället för ett matematiskt uttryck. Då behöver inte användaren uppmanas att omsluta svaret med citationstecken (“”).

Visa information

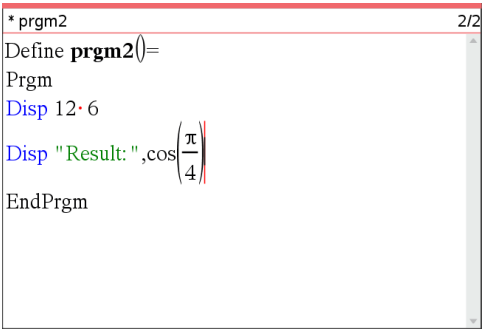
En funktion eller ett program som exekveras visar inte mellanliggande beräknade resultat såvida du inte inkluderar ett kommando om att de ska visas. Det är en viktig skillnad mellan att utföra en beräkning på inmatningsraden och att utföra den i en funktion eller i ett program.

Följande beräkningar visar exempelvis inte ett resultat i en funktion eller ett program (men de gör det från inmatningsraden).



Visa information i historiken

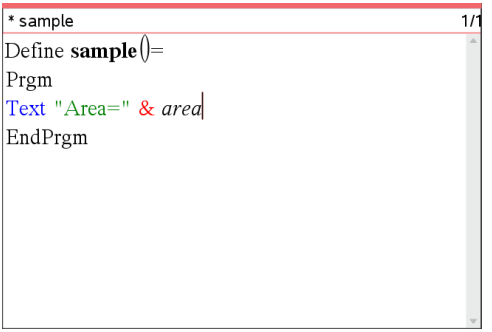
Du kan använda kommandot **Disp** i ett program eller en funktion för att visa information i historiken, inklusive mellanliggande resultat.



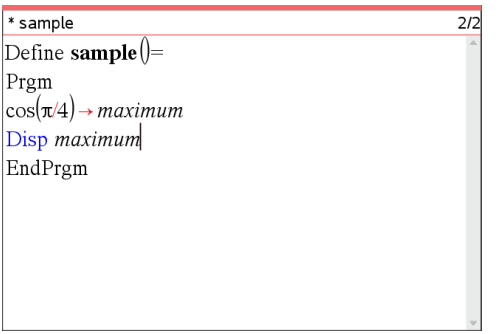
Visa information i en dialogruta

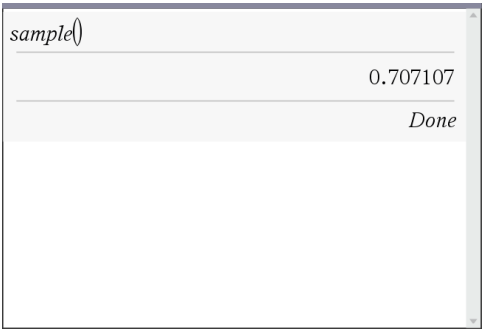
Du kan använda kommandot **Text** för att pausa ett pågående program och visa information i en dialogruta. Användaren väljer **OK** för att fortsätta eller **Cancel** för att stoppa programmet.

Du kan inte använda kommandot **Text** i en funktion.



Obs: Att visa ett resultat med **Disp** eller **Text** innebär inte att resultatet lagras. Om du förväntar dig att senare referera till ett resultat, lagra resultatet i en global variabel.



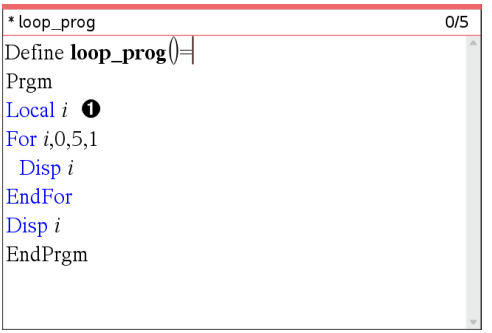


Använda lokala variabler

En lokal variabel är en temporär variabel som endast existerar medan en användardefinierad funktion beräknas eller ett användardefinierat program körs.

Exempel på en lokal variabel

Följande programsegment visar en **For...EndFor loop** (beskrivs längre fram). Variabeln *i* är slingans räknare. I de flesta fall används variabeln *i* endast medan programmet körs.



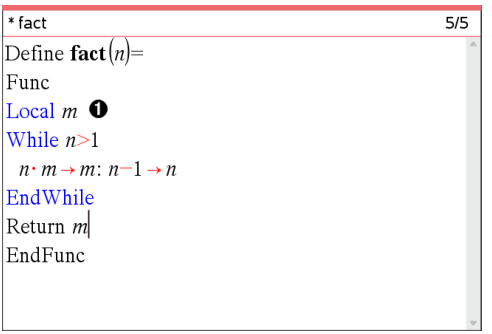
❶ Anger variabeln i som lokal.

Obs: När så är möjligt, ange som lokal varje variabel som endast används inom programmet och inte behöver vara tillgänglig när programmet stoppas.

Vad orsakar ett felmeddelande om odefinierad variabel?

Ett felmeddelande om **Undefined** variabel visas när du utvärderar en användardefinierad funktion eller kör ett användardefinierat program som refererar till en lokal variabel som inte har initialiserats (tilldelats ett värde).

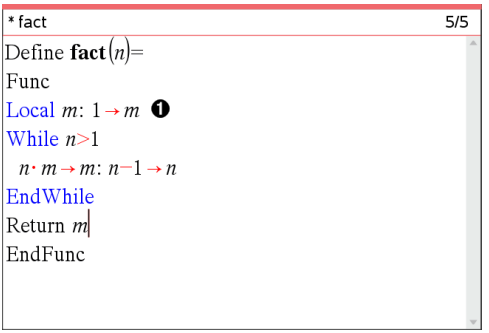
Till exempel:



❶ Den lokala variabeln m har inte tilldelats ett initialt värde.

Initialisera lokala variabler

Alla lokala variabler måste tilldelas ett initialt värde innan de kan refereras till.



❶ 1 lagras som det initiala värdet på m.

Obs (CAS): Funktioner och program kan inte använda en lokal variabel för att utföra symboliska beräkningar.

CAS: Utföra symboliska beräkningar

Om du vill att en funktion eller ett program skall utföra symboliska beräkningar måste du använda en global variabel i stället för en lokal. Du måste dock vara säker på att den globala variabeln inte redan finns utanför programmet. Följande metoder kan vara till hjälp.

- Referera till ett globalt variabelnamn, normalt med två eller flera tecken, som sannolikt inte finns utanför funktionen eller programmet.
- Inkludera **DelVar** inom ett program för att ta bort den globala variabeln, om den finns, innan du refererar till den. (**DelVar** tar inte bort låsta eller länkade variabler.)

Skillnader mellan funktioner och program

En funktion definierad i Programeditorn liknar de funktioner som är inbyggda i TI-Nspire™ programvara.

- Funktioner måste ge ett resultat som kan plottas eller matas in i en tabell. Program kan inte ge ett resultat.
- Du kan använda en funktion (men inte ett program) inom ett uttryck. Till exempel: **3 • func1(3)** is valid, but not **3 • prog1(3)**.
- Du kan endast köra program från applikationerna Räknare och anteckningar. Du kan dock utvärdera funktioner i Räknare, Anteckningar, Listor och kalkylblad, Grafer och geometri samt Data och statistik.
- En funktion kan referera till vilken variabel som helst, men den kan endast lagra ett värde till en lokal variabel. Program kan lagra både lokala och globala variabler.

  Obs: Argument som används för att överföra värden till en funktion behandlas automatiskt som lokala variabler. Om du vill lagra till andra variabler måste du ange dem som **Local** inifrån funktionen.

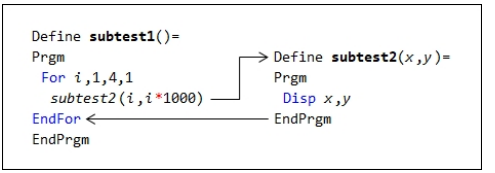
- En funktion kan inte anropa ett program som en subrutin, men den kan anropa en annan användardefinierad funktion.
- Du kan inte definiera ett program inom en funktion.
- En funktion kan definiera en lokal funktion, men inte en global funktion.

Anropa ett program från ett annat

Ett program kan anropa ett annat program som en subrutin. Subrutinen kan vara extern (ett separat program) eller intern (inkluderad i huvudprogrammet). Subrutiner är användbara när ett program behöver upprepa samma grupp av kommandon på flera olika ställen.

Anropa ett separat program

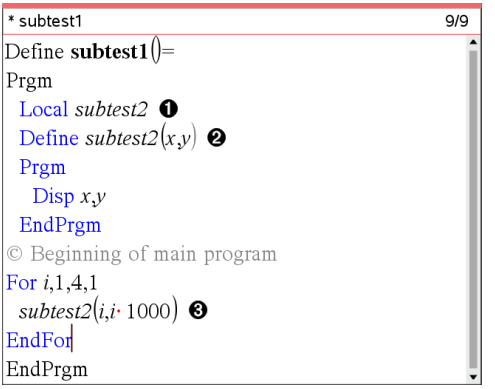
För att anropa ett separat program, använd samma syntax som du använder för att köra programmet från inmatningsraden.



Definiera och anropa en intern subrutin

För att definiera en intern subrutin, använd kommandot **Define** tillsammans med **Prgm...EndPrgm**. Eftersom en subrutin måste definieras innan den kan anropas är det god praxis att definiera subrutiner i början av huvudprogrammet.

En intern subrutin anropas och exekveras på samma sätt som ett separat program.



❶ Anger subrutinen som en lokal variabel.

❷ Definierar subrutinen.

❸ Anropar subrutinen.

Obs: Använd Programeditorns **Var**-meny för att mata in kommandona **Define** och **Prgm...EndPrgm**.

Anmärkningar om användningen av subrutiner

I slutet av en subrutin återgår exekveringen till det anropande programmet. För att när som helst gå ur en subrutin, använd **Return** utan argument.

En subrutin kan inte nå lokala variabler angivna i det anropande programmet. På liknande sätt kan det anropande programmet inte nå lokala variabler angivna i en subrutin.

Lbl-kommandon är lokala för programmen i vilka de är placerade. Ett **Goto**-kommando i det anropande programmet kan därför inte hoppa till ett läge i en subrutin eller vice versa.

Undvika cirkulära definitionsfel

När du utvärderar en användardefinierad funktion eller kör ett program kan du specificera ett argument som innehåller samma variabel som användes för att definiera funktionen eller skapa programmet. För att unvika cirkulära definitionsfel måste du dock tilldela ett värde för variabler som används för att utvärdera funktionen eller köra programmet. Till exempel:

$x+1 \rightarrow x$ ❶

– eller –

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

❶ Orsakar ett felmeddelande om **Circular definition error** om x eller i saknar värde. Felet uppstår inte om x eller i redan har tilldelats ett värde.

Kontrollera flödet i en funktion eller ett program

När du kör ett program eller utvärderar en funktion exekveras programraderna i sekventiell följd. Vissa kommandon ändrar dock programflödet. Till exempel:

- Kontrollstrukturer som till exempel, **If...EndIf**-kommandon använder ett villkorligt test för att bestämma vilken del av ett program som skall exekveras.
- Slingkommandon som till exempel, **For...EndFor** upprepar en grupp av kommandon.

Använda If, Lbl och Goto för att kontrollera programflöde

Kommandot **If** och flera **If...EndIf**-strukturer låter dig exekvera ett påstående eller ett block av påståenden villkorligt, dvs. baserat på resultatet av ett test (till exempel, x>5). **Lbl**- (märka) och **Goto**-kommandona låter dig hoppa från ett ställe till ett annat ställe i en funktion eller ett program.

Kommandot **If** och flera **If...EndIf**-strukturer finns på Programeditorns **Control**-meny.

När du infogar en struktur som till exempel, **If...Then...EndIf** infogas en mall vid markören. Markören placeras så att du kan skriva in ett villkorligt test.

If-kommandot

För att exekvera ett enda kommando när ett villkorligt test är sant, använd den allmänna formen:

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

❶ Exekveras endast om x>5, annars överhoppas.

❷ Visar alltid värdet på x.

I detta exempel måste du lagra ett värde till x innan du exekverar **If** -kommandot.

If...Then...EndIf-strukturer

För att exekvera en grupp av kommandon om ett villkorligt test är sant, använd strukturen:

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

❶ Exekveras endast om x>5.

❷ Visar värdet på:
2x om x>5
x om x≤5

Obs: EndIf markerar slutet på **Then**-blocket som exekveras om villkoret är sant.

If...Then...Else...EndIf-strukturer

För att exekvera en grupp av kommandon om ett villkorligt test är sant och en annan grupp om villkoret är falskt, använd denna struktur:

```
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 Exekveras endast om x>5.

❷ Exekveras endast om x≤5.

❸ Visar värdet på:
2x om x>5
5x om x≤5

If...Then...ElseIf...EndIf-strukturer

En mer komplex form av If-kommandot låter dig testa flera villkor. Anta att du vill att ett program skall testa ett av användaren angivet argument som betecknar ett av fyra alternativ.

För att testa varje alternativ (If Val=1, If Val=2, osv.), använd strukturen **If...Then...ElseIf...EndIf**.

Lbl- och Goto-kommandon

Du kan även kontrollera flödet genom att använda kommandona **Lbl** (märka) och **Goto**. Dessa kommandon finns på Programeditorns **Transfers**-meny.

Använd **Lbl**-kommandot för att märka (tilldela ett namn) ett visst ställe i funktionen eller programmet.

Lbl *labelName*	Namn som tilldelas detta ställe (använd samma konvention för namngivning som för ett variabelnamn).

Du kan sedan använda **Goto**-kommandot när som helst i funktionen eller programmet för att hoppa till det ställe som representeras av den specificerade etiketten.

Goto *labelName*	Specificerar vilket **Lbl**-kommando att hoppa till.

Eftersom ett **Goto**-kommando är ovillkorligt (det hoppar alltid till den specificerade etiketten) används det ofta med ett **If**-kommando så att du kan specificera ett villkorligt test. Till exempel:

```
If x>5
  Goto GT5 ❶
Disp x

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

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

❶ Om x>5, hoppar direkt till etikett GT5.

❷ För detta exempel måste programmet innehålla kommandon (till exempel, **Stop**) som förhindrar att **Lbl** GT5 exekveras om x≤5.

Använda slingor för att upprepa en grupp av kommandon

Använd en av slingstrukturerna för att successivt upprepa samma grupp av kommandon. Det finns flera typer av slingor. Varje typ ger dig ett visst sätt att gå ur slingan, baserat på ett villkorligt test.

Slingor och slingrelaterade kommandon finns på Programeditorns **Control**- och **Transfers**-menyer.

När du infogar en av slingstrukturerna infogas dess mall vid markören. Du kan sedan börja mata in de kommandon som skall exekveras inom slingan.

For...EndFor-slingor

En **For...EndFor**-slinga använder en räknare för att kontrollera antalet gånger slingan upprepas. Syntaxen för **For**-kommandot är:

Obs: Slutvärdet kan vara mindre än startvärdet, förutsatt att inkrementet är negativt.

For *variabel*, *start*, *slut* [, *inkrement*]
❶ ❷ ❸ ❹

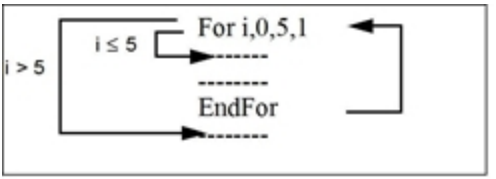
❶ *Variabel* används som räknare.

❷ Räknarvärdet som används första gången **For** exekveras.

❸ Går ur slingan när *variabel* överskrider detta värde.

❹ Läggs till i räknaren varje gång **For** exekveras (om detta valfria värde utelämnas är *inkrement* 1).

När **For** exekveras jämförs *variabel*-värdet med *slut*-värdet. Om *variabel*-värdet inte överskrider *slut*-värdet exekveras slingan. I annat fall hoppar kontrollen till det kommando som kommer efter **EndFor**.



Obs: For-kommandot inkrementerar automatiskt räknarvariabeln så att funktionen eller programmet kan gå ur slingan efter ett visst antal upprepningar.

I slutet av slingan (**EndFor**) hoppar kontrollen tillbaka till **For**-kommandot där variabeln inkrementeras och jämförs med *slut*-värdet.

Till exempel:

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

❶ Visar 0, 1, 2, 3, 4 och 5.

❷ Visar 6. När *variabel* inkrementeras till 6 exekveras inte slingan.

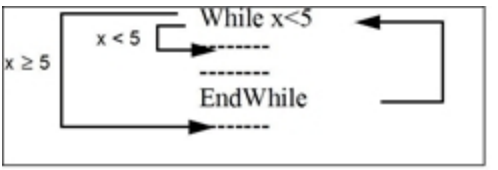
Obs: Du kan ange räknarvariabeln som lokal om den inte behöver sparas när funktionen eller programmet stoppas.

While...EndWhile-slingor

En **While...EndWhile**-slinga upprepar ett block av kommandon så länge ett specificerat villkor är sant. Syntaxen för **While**-kommandot är:

While *villkor*

När **While** exekveras utvärderas *villkor*. Om *villkor* är sant exekveras slingan. I annat fall hoppar kontrollen till det kommando som kommer efter **EndWhile**.



Obs: While-kommandot ändrar inte villkoret automatiskt. Du måste inkludera kommandon som låter funktionen eller programmet gå ur slingan.

I slutet av slingan (**EndWhile**) hoppar kontrollen tillbaka till **While**-kommandot där villkoret utvärderas på nytt.

För att exekvera slingan första gången måste villkoret initialt vara sant.

- Alla refererade variabler i villkoret måste ställas in före **While**-kommandot. (Du kan bygga in värdena i funktionen eller programmet eller också kan du uppmana användaren att mata in värdena.)
- Slingan måste innehålla kommandon som ändrar värdena i villkoret så att det till slut blir falskt. I annat fall är villkoret alltid sant och funktionen eller programmet kan inte gå ur slingan (kallas en oändlig slinga).

Till exempel:

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

❶ Ställer initialt in x.

❷ Visar 0, 1, 2, 3 och 4.

❸ Inkrementerar x.

❹ Visar 5. När x inkrementeras till 5 exekveras inte slingan.

Loop...EndLoop-slingor

En **Loop...EndLoop** skapar en oändlig slinga som upprepas i all oändlighet. **Loop**-kommandot har inga argument.

Normalt infogar du kommandon i slingan som låter programmet gå ur slingan. Kommandon som ofta används är: **If**, **Exit**, **Goto** och **Lbl** (märka). Till exempel:

❶ Ett **If**-kommando kontrollerar villkoret.

❷ Går ur slingan och hoppar hit när x inkrementeras till 6.

Obs: Exit-kommandot går ur den aktuella slingan.

I detta exempel kan **If**-kommandot finnas var som helst i slingan.

När If-kommandot är:	**Så:**
I början av slingan	Exekveras slingan endast om villkoret är sant.
I slutet av slingan	Exekveras slingan minst en gång och upprepas endast om villkoret är sant.

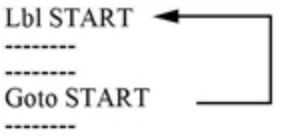
If-kommandot kan även använda ett **Goto**-kommando för att överföra programkontroll till ett specificerat **Lbl**-kommando (märka).

Upprepa en slinga omedelbart

Kommandot **Cycle** överför omedelbart programkontroll till nästa upprepning av en slinga (innan den aktuella upprepningen är slutförd). Detta kommando fungerar med **For...EndFor**, **While...EndWhile** och **Loop...EndLoop**.

Lbl- och Goto-slingor

Trots att kommandona **Lbl** (märka) och **Goto** i egentlig mening inte är slingkommandon kan de användas för att skapa en oändlig slinga. Till exempel:



I likhet med **Loop...EndLoop** skall slingan innehålla kommandon som låter funktionen eller programmet gå ur slingan.

Ändra lägesinställningar

Funktioner och program kan använda funktionen **setMode()** för att tillfälligt ställa in specifika beräknings- eller resultatlägen. Programeditorns **Mode**-meny gör det enkelt att skriva in rätt syntax utan att du behöver komma ihåg numeriska koder.

Obs: Lägesändringar som görs inom en funktions- eller programdefinition har ingen effekt utanför funktionen eller programmet.

Ställa in ett läge

1. Placera markören där du vill infoga **setMode**-funktionen.
2. På menyn **Läge**, välj det läge som ska ändras och välj den nya inställningen.

   Den rätta syntaxen infogas vid markören. Till exempel:

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

Avlusa program- och hanteringsfel

När du har skrivit en funktion eller ett program kan du använda flera metoder för att söka och korrigera fel. Du kan också bygga in ett felhanterande kommando i själva funktionen eller programmet.

Om funktionen eller programmet låter användaren välja mellan flera alternativ, var noga med att köra funktionen/programmet och testa varje alternativ.

Metoder för avlusning

Meddelanden om bearbetningsfel kan lokalisera syntaxfel, men inte fel i programlogik. Följande metoder kan vara till hjälp.

- Infoga tillfälligt **Disp**-kommandon för att visa värden på kritiska variabler.
- För att verifiera att en slinga exekveras rätt antal gånger, använd **Disp** för att visa räknarvariabeln eller värdena i det villkorliga testet.
- För att verifiera att en subrutin exekveras, använd **Disp** för att visa meddelanden som till exempel, “Entering subroutine” och “Exiting subroutine” i början och slutet av subrutinen.
- För att stoppa ett program eller en funktion manuellt,
  - Windows®: Håll ned **F12** och tryck på **Enter** upprepade gånger.
  - Macintosh®: Håll ned **F5** och tryck på **Enter** upprepade gånger.
  - Handenhet: Håll ned knappen [on] och tryck på [enter] upprepade gånger.

Felhanteringskommandon

Kommando	Beskrivning
Try...EndTry	Definierar ett block som låter en funktion eller ett program exekvera ett kommando och, vid behov, återhämta sig från ett fel som kommandot genererar.

Kommando	Beskrivning
ClrErr	Rensar felstatusen och ställer in systemvariabeln *errCode* på noll. Om du vill se ett exempel på användningen av *errCode* kan du läsa om kommandot **Try** i *Referensguiden*.
PassErr	Flyttar ett fel till nästa nivå i **Try...EndTry**-blocket.

Allmän information

Hjälp-funktion online

education.ti.com/eguide

Välj ditt land för ytterligare produktinformation.

Kontakta TI support

education.ti.com/ti-cares

Välj ditt land för teknisk och andra supportresurser.

Service- och garanti-information

education.ti.com/warranty

Välj ditt land för information om garantins längd och villkor eller om produkttjänsten.

Begränsad garanti. Denna garanti påverkar inte dina lagstadgade rättigheter.

Texas Instruments Incorporated

12500 TI Blvd.

Dallas, TX 75243