Kapitel 4: Loopar

Övning 3: Loop….Endloop

I denna aktivitet kommer du att lära dig att arbeta med den mångsidiga Loop…EndLoop-strukturen.

Syfte:

- Use the **Loop…EndLoop** structure
- Use the **Exit** statement to get out of a loop

Lärarkommentarer: De flesta mer traditionella programmerings-instruktörer gillar inte att använda Loop... EndLoop-struktur eftersom det främjar programmeringsovanor som leder till s.k. ”spagetti-kod”. (program som hoppar runt överallt med hjälp av GoTo-satser). Loop... EndLoop-strukturen tas upp här med avsikten att den adresseras riktigt: använd endast en Exit-sats och använd inte GoTo satser under några omständigheter!

En fördel med Loop... EndLoop strukturen är *möjligheten* med flera **Exit-punkter**, men detta skulle nog vara sällsynt och det finns alltid en lösning för att undvika denna situation. Exit-satsen tvingar programkontrollen att processa den första satsen efter EndLoop så det finns inget missförstånd om hur programflödet leds.

Hur fungerar egentligen Loop…EndLoop?

Loop…EndLoop strukturen ger möjlighet till en mer flexibel loop-struktur. Den upprepar exekvering av satser i loopen. Observera att loopen kommer att köras i all oändlighet, såvida inte en Exit-instruktion exekveras någonstans inuti loopen.

Utan en **Exit**-sats kommer loopen att vara oändlig.

Loop…EndLoop strukturen processas åtminstone en gång eftersom det inte finns något villkor som ska uppfyllas

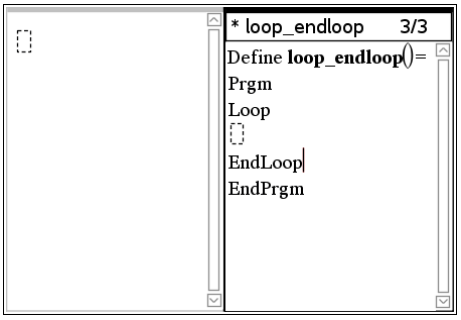
Exit tvingar programmet att processa satsen precis efter **EndLoop.**

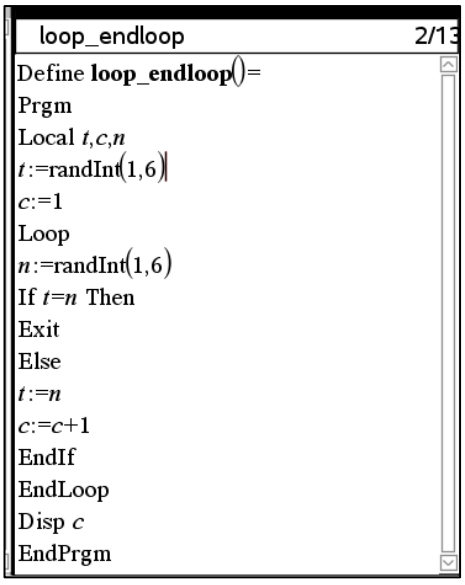
Exempel: Hur många slumptal från 1 till 6 kan du alstra innan två på varandra följande värden är lika?

Notera i programmet till höger hur **Loop…EndLoop** används med en *villkorlig* **Exit**-sats. När *randInt*(1,6) alstrar två *konsekutiva* (på varandra följande) identiska värden så avslutas loopen och processar Disp-satsen i slutet av programmet. **Exit** hittar du i programeditorn i menyn *Överföringar*.

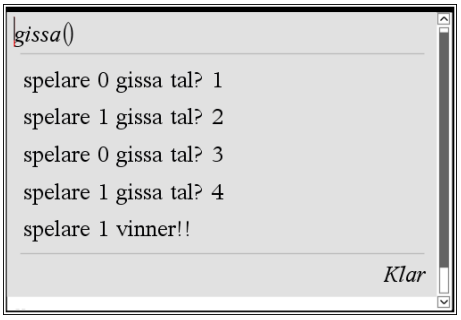
Samma effekt kan uppnås med en *While*-loop.

Syntax: Man kan ha två satser på samma rad i programmet genom att separera satserna med ett kolon (:).





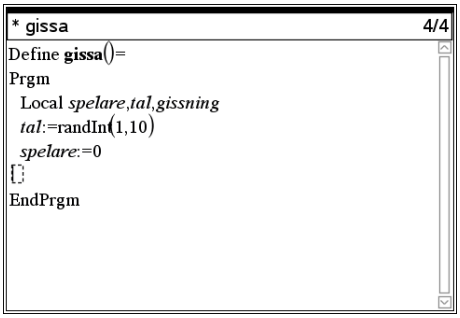
Lärarkommentar: En fördel med **Loop…EndLoop-**strukturen är möjligheten av flera **Exit**-punkter. Vi rekommenderar dock att eleverna undviker detta tills de har skaffat sig större programmeringsskicklighet.



Program: Gissa mitt tal

1. För att demonstrera användningen av Loop... EndLoop, ska vi nu konstruera ett spel för två spelare där det gäller att gissa ett slumptal mellan 1 och 10. När en spelare gissar rätt tal så avslutas programmet med ett meddelande om vilken spelare som vunnit. Till höger visar vi av resultatet från en programkörning. Vi ska nu komma igång med ett sådant program som vi döper till **gissa**.

2. Vi börjar med att identifiera tre variabler: ”spelare nr”(*spelare*), slumptalet (*tal*) and spelarens gissning (*gissning*).



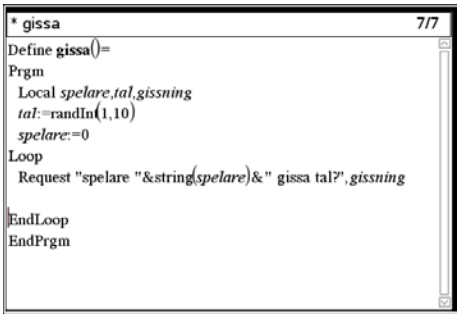
Datorn alstrar ett heltaligt slumptal mellan 1 och 10. Vi börjar med spelare nr 0. Detta gör det enkelt att ändra spelarens nummer, som vi ska visa senare.

Lärarkommentar: Kom ihåg att använda lokala variabler eftersom de annars kan störa andra aktiviteter i samma *problem*. Ett dokument kan ju bestå av ett eller flera problem.

3. Be spelaren att mata in en gissning. Den färdiga satsen blir:

   Request "spelare " & string(*spelare*) & " gissa?", *gissning*

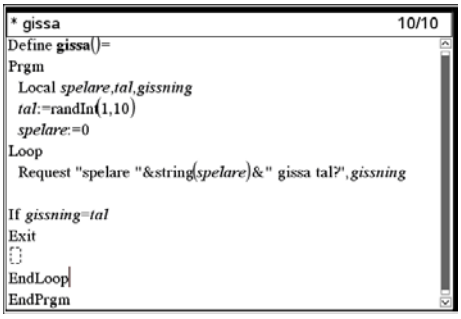
   & används för *sammanfogning* av strängar. Inmatningsdelen av en **Request**-sats måste vara en sträng så variabeln *spelare* omvandlas till en sträng med **string( )**-funktionen.



Lärarkommentar: *Sammanfogning* gör att man kan sätta ihop två strängar till en. Tecknen i den andra strängen läggs till i slutet av den första strängen.
Du måste använda string( )-funktionen för att omvandla en numerisk variabel till en strängvärde innan du sammanfogar den med andra strängar.

4. Därefter konstruerar man ett **Exit**-villkor. När en spelare gissar rätt tal så kommer loopen av avlutas. Här använder vi den enkel If-sats:

   If gissning=tal
   Exit

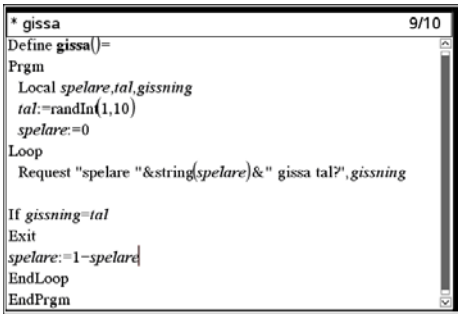


Tänk på att **If** utan **Then** processar nästa sats när villkoret är sant. I annat fall hoppas nästa sats över.

5. För att växla spelare kan man använda följande listiga sats:

   spelare*:= 1 – *spelare

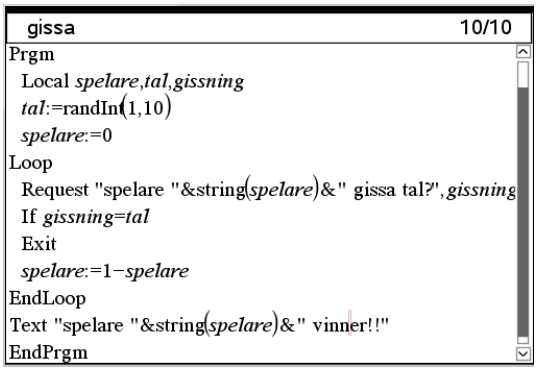
Denna sats byter värdet hos variabeln *spelare* från 0 till 1 och tvärtom från 1 till 0.



6. Till slut så lägger vi efter loopen till en sats som meddelar vem som vunnit.

   Text “spelare” & string(spelare) & “vinner!!”

Tips: Om du inte vill att spelarna ska benämnas ”spelare 0 och spelare 1 så kan man lägga till 1 till variabeln *spelare* i satsen här och i **Request**-satsen. Användaren ser 1 eller 2 även om datorn använder 0 och 1 för spelarna.



7. Innan du kör programmet se till att spara dokumentet. Du kan t.ex. fastna i en oändlig loop! Så fort programmet har startats så måste du fortsätta tills du har en vinnare.

Lärarkommentar: När man skriver program är det viktigt att med jämna intervall spara dokumentet där programmet finns. Om något går fel kan man annars förlora hela arbetet.