

Summan av tal som följer ett mönster

Summan av tal som följer ett mönster

Om du ska räkna ut $3 + 3 + 3 + 3 + 3 + 3$ så summerar du naturligtvis inte term för term utan du ser att det är 6 lika tal och då blir ju summan $3 \cdot 6 = 18$.

Men hur gör vi om termerna inte lika men följer ett mönster?

Om *skillnaden* mellan termerna är konstant är summan *aritmetisk*. Om *kvoten* mellan termerna är konstant är summan *geometrisk*.

Vi ska nu jobba med två program som beräknar sådana här summor och vi ska också titta på de verktyg som finns hos TI-Nspire för att arbeta med det man kallar talföljder.

På nästa sida har vi nu ett program som räknar ut summor hos talföljder som är geometriska. Det gäller att nästa tal i talföljden följer från tidigare tal enligt en bestämd regel. Sedan har man ett tal som sätter igång talföljden.

Ett exempel på en rekursiv geometrisk talföljd får man genom att t.ex. börja med talet 1 (startvärde) och sedan *halvera* talet varje gång.

Man skulle då kunna skriva **Nuvarande tal= föregående tal·1/2**

I detta fall blir det då talföljden 1, 1/2, 1/4, 1/8... Sedan ska summera denna följd av tal.

Det är vad följande program gör. Man matar in ett startvärde, kvoten mellan talen och hur många termer som ska summeras.

Lite förklaringar till programmet på nästa sida.

På sid 3 ger vi en del korta förklaringar till kommandon som dyker upp i programmet.

Utförliga förklaringar hittar man i de aktiviteter som hör till **10 minutes of Code**.

Koda med TI: TI-Nspire™ CX

Korta aktiviteter som ger eleverna praktisk erfarenhet med tekniska hjälpmedel, utvecklar kritiskt tänkande och fördjupar förståelsen av begrepp i matematik och programmering.

10 Minutes of Code

- Kapitel 1: Grundläggande programmering
- Kapitel 2: Tilldela värden till variabler
- Kapitel 3: Villkorssatser
- Kapitel 4: Loopar
- Kapitel 5: Listor, grafer och dynamiska program

Local

I kapitel 2, Övning 2 kan du läsa mer om hur **lokala** variabler kan användas.

Kapitel 2: Tilldela värden till variabler

I denna andra aktivitet för kapitel 2 kommer du att lära dig att deklarerar **lokala** variabler i ett program.

Övning 2: Lokala variabler

Syfte:

- Förstå behovet av lokala variabler
- Använda lokala variabler i program

Börja med att öppna dokumentet som innehåller programmet heron som du skrev i den förra aktiviteten. Se skärmbilden till höger.

Kom ihåg att i detta program så skapas variabeln `s` oavsiktligt i problemet i dokumentet. Det är ingenting vi önskar och vi visar nu hur det går till att förhindra detta.

Lägg nu till programsatsen

Local s

Local finner du i programeditorn under 3:Definiera variabler. Se skärmbilden till höger.

När du är klar så ska du "Kontrollera Syntax och lagra" Se meny i programeditorn. Det finns också ett snabbval för denna åtgärd, nämligen Ctrl B.

Gå nu över till Räkna-appen till vänster och skriv kommandot **DelVar s** för att ta bort variabeln `s`. Du hittar kommandot i menyen under **Åtgärder**.

DelVar s **Klar**

Kör nu programmet heron igen.

Om du trycker på **Vär**-knappen ser du att den enda variabel som finns är själva programmet heron.

Vad händer?

När du deklarerar (eller anger) att en variabel är **Lokal** så informerar den TI-Nspire™ CX att skapa en variabel precis när den exekverar programmet och ta bort den när programmet slutar så att den inte längre existerar i problemet där programmet finns. (Ett dokument består av ett eller flera problem.) Detta eliminerar problemet med att variabeln skapas och sedan finns kvar. Om variabeln redan finns så påverkar inte en lokal variabel eftersom programmet "tillverkar" sin egen temporära variabel.

Request

I kapitel 3, Övning 1 tar vi upp hur man använder kommandot Request.

Kapitel 3: Villkorssatser

I denna första lektion för kapitel 3 kommer du att lära dig om kommandot **Request**, som används vid inmatning från användaren medan programmet körs. Vi tar också upp strängar och den mest grundläggande villkorssatsen, nämligen **If**.

Övning 1: Request och If

Syfte:

- Använda **Request** och **RequestStr** vid inmatning.
- Undersöka strängvariabler och sammanfogning.
- Skriva satser där man använder **If** och villkor.

Lärarkommentar: Denna aktivitet innehåller flera olika saker som egentligen inte hör ihop: inmatningar, strängar & sammanfogningar, villkor och den enkla If-satsen. Programexemplet är inte alls komplicerat och det demonstrerar alla de nämnda funktionerna.

I några av de skärmar som dyker upp i högermarginalen visas programeditorn på en egen sida. Om man har en sida med programeditorn och Räkna-appen på en delad sida så kan man få dem på egna sidor genom att gå till

Redigera → Sidlayout och därefter välja **Dela Grupp**.

Man kan också från denna meny **Gruppera** på samma sida. Snabbkommandona för dessa åtgärder är Ctrl 6 och Ctrl 4.

Översikt

Hittills har vi bara hittills har vi bara kunnat ge ett program eller en funktion värden med hjälp av argument. Hos TI-Nspire™ CX finns det två liknande satser som tillåter dig att mata in värden till programmet medan det körs. De kallas för inmatningskommandon:

- Request** "meddelande", variabel (för numerisk inmatning)
- RequestStr** "meddelande", variabel (för inmatning av sträng)

Du hittar dessa satser i **I/O**-menyn i programeditorn. Se figur.

Typer av variabelinmatningar

- En numerisk variabel kan innehålla ett reellt eller komplext tal, en lista eller till och med en matris. Den kan användas i algebraiska uttryck och dess värde används vid beräkningarna av uttrycken.
- En strängvariabel kan innehålla text som kan bestå av bokstäver, siffror och de flesta skiljetecken. Den kan naturligtvis användas i algebraiska uttryck. Observera att tal är tillåtna i strängarna.
- "Meddelandet" kallas "prompt". Det beskriver för användaren vad som förväntas att inmatas.
- Variabeln kan vara en bokstav eller ett ord som inte är ett reserverat sådant. Man känner igen reserverade ord genom att ordet ändras till normal stil från kursiv stil när man skriver det.

For---EndFor

For *variabel, startvärde,slutvärde* [, *steg*]
kodblock

EndFor

Exekverar ett *kodblock* från och med *Variabel* lika med *startvärde* och ökar sedan *Variabel* upprepade gånger i *Steg* och stoppar när värdet överstiger *slutvärde*. Om *Steg* inte anges har det värdet 1.

Man kan läsa mer om användning av For ... EndFor i kapitel 4, Övning 1.

Kapitel 4: Loopar **Övning 1: For-loopar**

I denna aktivitet kommer du att lära dig hur begreppet looping fungerar i programmering och närmare undersöka For-loopen.

Syfte:

- Beskriva hur looping fungerar i programmering
- Konstruera program där man använder For...EndFor

Lärarkommentar: Det finns tre grundläggande loopar i TI-Nspire TI-Basic: For, While, och Loop. En loopstruktur ger ett program förmåga att processa en uppsättning av satsar om och om igen, antingen upprepning över en sekvens av värden (precis som i For-loopen) eller tills ett speciellt villkor är uppfyllt (eller inte) som i While och Loop. Aktiviteterna i kapitel 4 introducerar var och en av dessa strukturer. Program kan bli komplicerade eftersom det ofta är nödvändigt att blanda in alla kontrollstrukturer (If-sats och loopar) i ett program för att kunna arbeta med mer komplexa algoritmer. Det är ju detta som gör programmering så spännande och intressant ... och roligt!


Om Loopar
Programmeringsspråket TI-Basic har förmågan att bearbeta en uppsättning programsatser om och om igen. Denna upprepning av uttalanden kallas **looping**. De tre loop-strukturer du lär dig i detta kapitel är nås genom att välja kontrollmenyn i programeditorn. Se skärmbild till höger. While... och Loop... strukturer kommer att utforskas i senare aktiviteter i detta kapitel.

For...EndFor
For... loopen används för att processa en aritmetisk sekvens av värden. Det kallas för *iteration*. Genom att välja For...EndFor-satsen från kontrollmenyn får du tillgång till de nödvändiga komponenterna som behövs för att bygga resten av strukturen:

```
For i, start, stop, step
```

EndFor
Kommatecknet efter ordet For indikerar att du behöver lägga till fyra poster:

```
For i, 1, n, 1
```



Så här ser nu programmet ut för att beräkna delsummor till geometriska talföljder.

talfoljd_geo 1/11

```

Define talfoljd_geo ()=
Prgm
Local i,summa_av_varden,termer
Request "första termen",start
Request "kvot",kvot
Request "antal termer",termer
totalt:=start
summa_av_varden:=start
For i,2,termer,1
summa_av_varden:=summa_av_varden·kvot
totalt:=totalt+summa_av_varden
Disp "term ",i,"=",summa_av_varden,"och totalsumman=",totalt
EndFor
EndPrgm
    
```

Med programmet så beräknar vi alltså varje term och hur stor totalsumman blir efter varje steg.

Här är en körning av programmet där första termen är 1 och kvoten 1/2. Vilket tal närmar sig summan om antalet termer ökar?

talfoljd()

första termen 1
kvot 1/2
antal termer 5

term 2 = $\frac{1}{2}$ och totalsumman = $\frac{3}{2}$
term 3 = $\frac{1}{4}$ och totalsumman = $\frac{7}{4}$
term 4 = $\frac{1}{8}$ och totalsumman = $\frac{15}{8}$
term 5 = $\frac{1}{16}$ och totalsumman = $\frac{31}{16}$

Klar

Här en andra körning som är ett åskådliggörande av händelsen "insättning av 1000 kr varje år under 5 år med 10 % ränta."

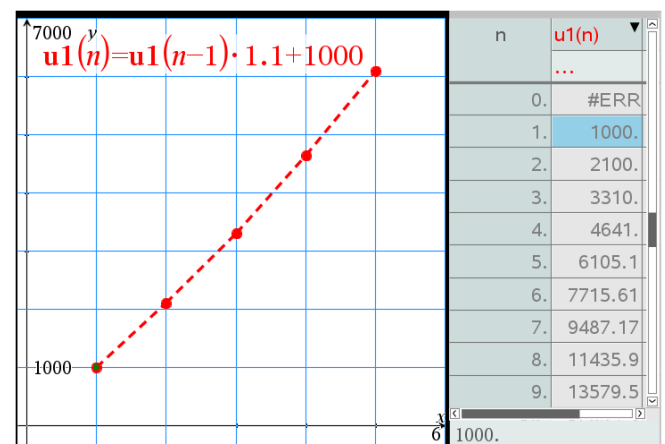
talfoljd()

första termen 1000
kvot 1.10
antal termer 5

term 2 = 1100. och totalsumman = 2100.
term 3 = 1210. och totalsumman = 3310.
term 4 = 1331. och totalsumman = 4641.
term 5 = 1464.1 och totalsumman = 6105.1

Klar

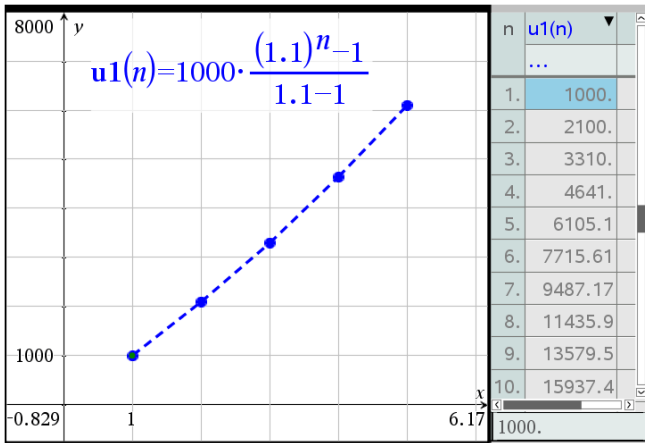
Hos TI-Nspire kan man i graf-appen arbeta med talföljder på olika sätt. Vi visar här det rekursiva sättet eftersom det påminner om hur man programmerar.



Summan av den geometriska talföljden kan också beräknas med en explicit formel

$$\text{startvärde} \cdot \frac{\text{kvot}^n - 1}{\text{kvot} - 1} \text{ (för kvot} \neq 1\text{).}$$

Se nästa sida.



I TI-Nspire-dokumentet visas också ett exempel där man betalar av en skuld på 100 000 kr med 6 % ränta och en årlig avbetalning på 10 000 kr (ränta+ amortering).

Rekursiva beräkningar kan man göra i kalkylarket också.

$$a_2 = a_1 \cdot 1,1 + 1000$$

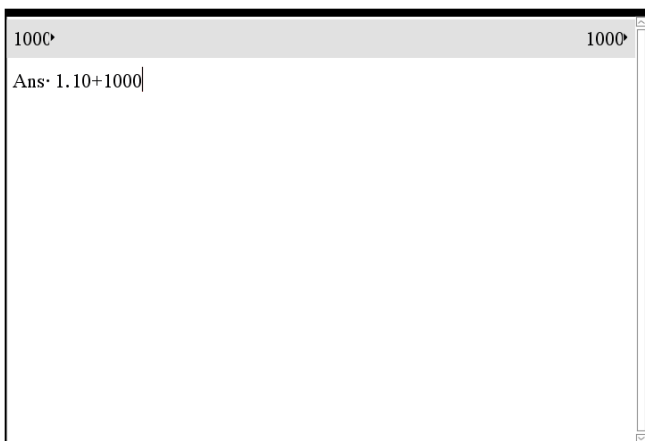
$$a_3 = a_2 \cdot 1,1 + 1000$$

osv.

A	B	C
1	1000	
2	2100.	
3	3310.	
4	4641.	
5	6105.1	
6	7715.61	
7	9487.17	
8	11435.9	
9	13579.5	
10	15937.4	

Man kan också göra beräkningar i kalkylarket. Klicka på cell a2 och så förstår du.

Det finns faktiskt ett tredje sätt. Gå till Räknare-appen och skriv in 1000 och tryck enter. Skriv sedan $\cdot 1.10 + 1000$. Då skrivs automatiskt Ans in före gångertecknet. Ans är resultatet av den sista beräkningen.



Tryck nu på enter igen upprepade gånger.

1000	1000
1000 · 1.1+1000	2100.
2100 · 1.1+1000	3310.
3310 · 1.1+1000	4641.
4641 · 1.1+1000	6105.1
6105.1 · 1.1+1000	7715.61
7715.61 · 1.1+1000	9487.17

Vi får summor beräknade precis som i kalkylarket.

Problem 2

Här har vi istället *aritmetiska* talföljder. Nästan samma program. Här har vi istället raden

$Summa_av_värden := summa_av_värden + differens$

```

aritmisk_talf 8/11
Define aritmisk_talf()=
Prgm
Local i,summa_av_värden,termer
Request "första termen",start
Request "differens",differens
Request "antal termer",termer
totalt:=start
summa_av_värden:=start
For i,2,termer,1
summa_av_värden:=summa_av_värden+differens
totalt:=totalt+summa_av_värden
Disp "term ",i,"=",summa_av_värden," och totalsumman=",totalt
EndFor
EndPrgm

```

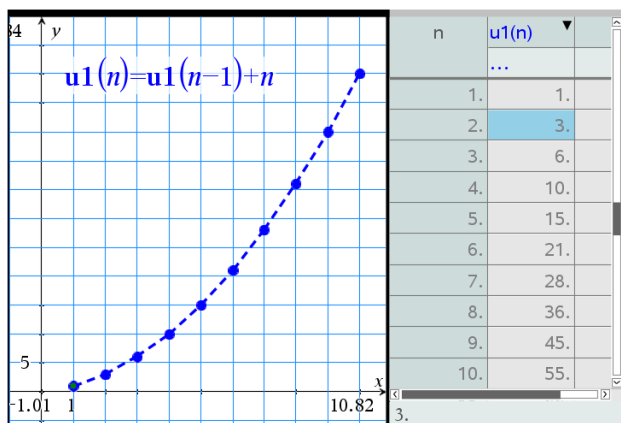
Aritmetiska talföljden 1, 2, 3

```

aritmisk_talf()
första termen 1
differens 1
antal termer 10
term 2 = 2 och totalsumman= 3
term 3 = 3 och totalsumman= 6
term 4 = 4 och totalsumman= 10
term 5 = 5 och totalsumman= 15
term 6 = 6 och totalsumman= 21
term 7 = 7 och totalsumman= 28
term 8 = 8 och totalsumman= 36
term 9 = 9 och totalsumman= 45
term 10 = 10 och totalsumman= 55

```

På nästa sida visar vi hur man tar fram summorna med TI-Nspire's grafverktyg.

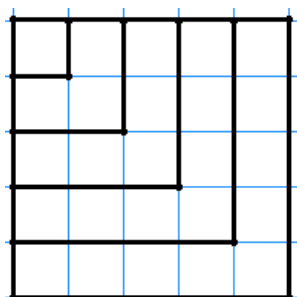


Här en programkörning där differensen är 2., dvs vi ska summera termerna i talföljden 1, 3, 5, 7 ...

The figure shows a TI-Nspire calculator screen titled "aritmetisk_talf.". It displays the following information:

- första termen 1
- differens 2
- antal termer 10
- term 2 = 3 och totalsumman = 4
- term 3 = 5 och totalsumman = 9
- term 4 = 7 och totalsumman = 16
- term 5 = 9 och totalsumman = 25
- term 6 = 11 och totalsumman = 36
- term 7 = 13 och totalsumman = 49
- term 8 = 15 och totalsumman = 64
- term 9 = 17 och totalsumman = 81
- term 10 = 19 och totalsumman = 100

Vi ser att delsummorna blir 1, 4, 9 dvs. kvadraten på ordningsnumret för termer. Ett geometriskt bevis visas nedan. Titta på följande figur:



Figuren visar att de n första udda talen beskriver en uppdelning av en kvadrat med sidan n . Kvadratens area är n^2 .

The figure shows a TI-Nspire calculator screen with the following text:

Du kan naturligtvis använda de inbyggda verktygen hos TI-Nspire:

$\text{seq}(2 \cdot k - 1, k, 1, 10) \rightarrow \{1, 3, 5, 7, 9, 11, 13, 15, 17, 19\}$

$\text{sum}(\text{seq}(2 \cdot k - 1, k, 1, 10)) \rightarrow 100$

Below this, two summation formulas are shown:

$$\sum_{k=1}^{10} (2 \cdot k - 1) \rightarrow 100 \quad \sum_{k=1}^n (2 \cdot k - 1) \rightarrow n^2$$