Kapitel 3: Starta programmering på riktigt

Övning 1: Funktioner och loopar

I den första lektionen i kapitel 3 använder du dina kunskaper om algoritmer och Pythonspråket för att lära dig:

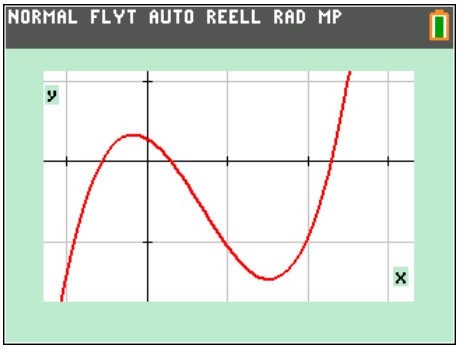
- Intervallhalveringsmetoden i algebra och i sannolikhetslära
- Söka efter lösningar till en tredjegradsekvation

Syfte:

- Implementera en avgränsad while-loop
- Visa på två metoder för att lösa ett problem

Vi skall i denna övning också lösa en ekvation, dvs. finna noll-ställen till funktionen. Vi ska titta på något som kallas intervallhalve-ringsmetoden. Som exempel har vi valt tredjegradsfunktionen

$$f(x)=0{,}03x^3-0{,}3x^2-0{,}6x+1{,}45$$



Denna ekvation kan vi inte lösa exakt med elementära metoder.

Vi börjar med att rita grafen till funktionen för att få en uppfattning om var nollställena är placerade. En tredjegradsfunktion har i all-mänhet tre nollställen.

Mellan varje skalstreck är det 5 längdenheter. Vi väljer att titta på det nollställe som ligger längst till höger. Nollstället ligger mellan 10 och 15.

För att förklara den metod vi kommer att använda så visar vi samma metod i ett enklare sammanhang:

Din kompis ber dig gissa en sida i en tjock bok på 1024 sidor. Efter varje gissning så kan din kompis bara svara lägre eller högre. Hur många gissningar behöver du maximalt göra för att hamna på rätt sida? Vi tar ett exempel. Vi säger att det riktiga svaret är 273, som du naturligtvis inte vet om.

1. Du har tänkt igenom det här och svarar hälften av 1024, dvs 512.
   Det är inte rätt utan svaret ligger *tidigare*, dvs i intervallet 1-511.
2. Du gissar nu hälften av 512, dvs 256. Det är inte rätt utan svaret ligger *senare*, dvs i intervallet 257-511.
3. Du gissar nu mitt i detta intervall, dvs 384. (257+511)/2=384). Det är inte heller rätt utan svaret ligger tidigare, dvs i intervallet 257-383. Mitt i är då (257+383)/2=320. De fortsatta gissningarna här blir
   288, 272 osv.
4. Så här håller du på, och du tar hela tiden intervall som är hälften av det föregående, och du kommer allt närmare det riktiga svaret. Nu kan du förstås ha tur och gissa rätt bara efter några gissningar.

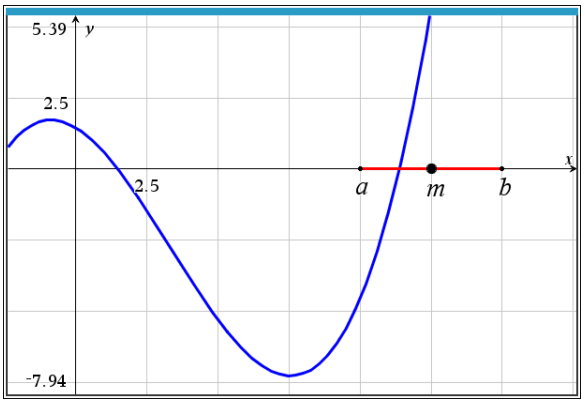
Detta resonemang med intervallhalveringar ger att man maximalt behöver 10 gissningar eftersom $2^{10} = 1024$. Man kan också skriva att $\left(\frac{1}{2}\right)^{10} = \frac{1}{1024}$.

Beskrivning av intervallhalveringsmetoden

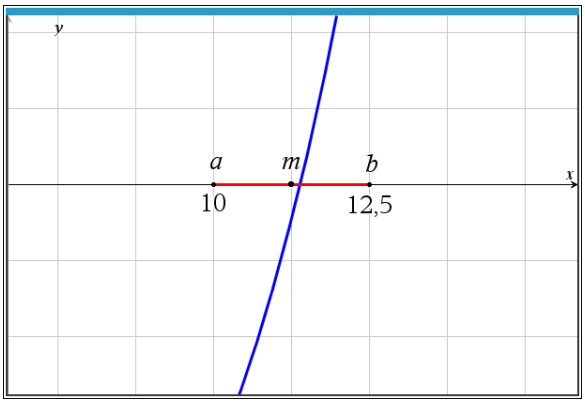
Nu ska vi bli lite mer formella och försöka göra en bra algoritm av detta som passar till vårt problem med funktionens nollställe.

Metoden bygger på följande resonemang: Om en funktion växlar tecken i intervallet [a, b] och om funktionen är kontinuerlig så är funktionen noll någonstans i intervallet. Man har då minst ett nollställe i intervallet. Delar man nu intervallet i mitten så att man får två nya intervall. Då kommer vi fortfarande ha teckenväxling i något av de nya intervallen. Detta intervall kommer då att innehålla minst ett nollställe och vi fortsätter med detta intervall och upprepar resonemanget.

Vi förutsätter också att funktionen f(x) är kontinuerlig. Man börjar med ett intervall [a, b] där f(x) växlar tecken, dvs. där $f(a)\cdot f(b) < 0$.



Eftersom $f(x)$ är kontinuerlig så finns det minst ett nollställe i intervallet [a, b]. Man bildar nu *mittpunkten* m = (a+b()/2 och delar alltså intervallet i två lika långa delintervall [a, m] och [m, b]. Se figuren. Det delintervall där funktionen växlar tecken är det man är intresserad av. Man behåller alltså [a, m] om $f(a)\cdot f(m) < 0$, annars behåller vi [m, b]. I detta fall behåller vi alltså [a, m]. Se figur. Sedan går man vidare och delar detta intervall på mitten igen.

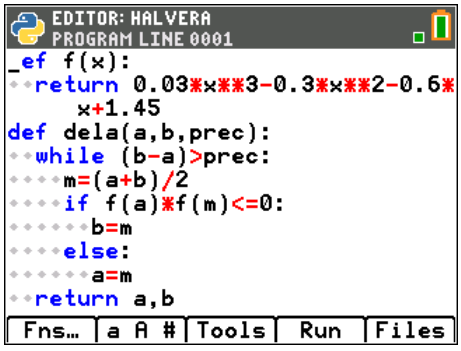


Vi gör nu en förstoring och tittar nu på intervallet [10, 12.5] och vi ser att nollstället finns i den högra halvan, dvs i intervallet [11,25, 12,5]. Så här håller man på tills man har ett intervall som är tillräckligt litet och man har uppnått önskad precision i beräkningarna.

Förberedelse för skapandet av ett skript

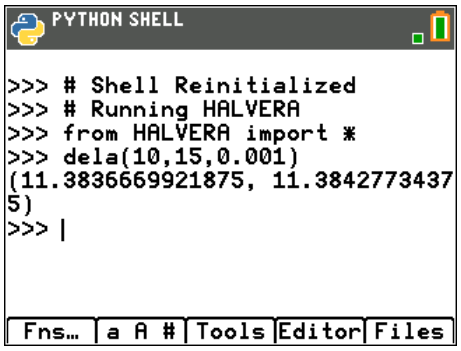
1. Först ser man att nollstället finns mellan 10 och 15.
2. Man tar nu medelvärdet av 10 och 15 och ser nästa gång att nollstället ligger mellan 10 och 12,5.
3. Nu tar man medelvärdet av 10 och 12,5 och då ser man (nedre figuren) att nollstället ligger mellan 11,25 och 12,5.
4. Så här fortsätter man med allt mindre och mindre intervall.

Efter den här ganska långa genomgången kan man börja skriva in sitt skript.



Först definierar man sin funktion och sedan definierar man en funktion med argumenten *a*, *b* och *prec*. *a* och *b* är gränserna i intervallen och *prec* är precisionen, dvs differensen mellan *a* och *b*.

Så länge som (*while*) avståndet b-a är större än den precision man vill ha så utförs beräkningar av medelvärde och sedan kommer *if …else*-sats med beräkning av vilket tecken (positivt eller negativt värde) som f(a)*f(m) har. Är det negativt så har vi en teckenväxling och nollstället ligger i det vänstra delintervallet. Är det positivt så ligger nollstället i det högra delintervallet.



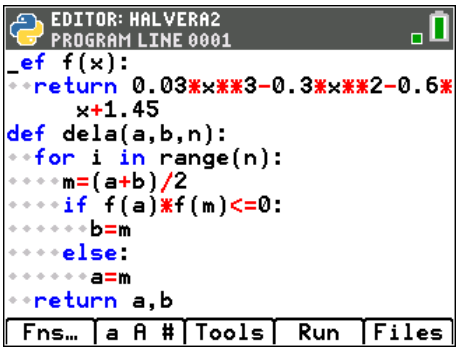
Tips för inskrivning: Olikhetstecken kommer du åt genom att trycka 2nd math. while finns i kontrollmenyn. Tryck f1 (Fns..) och sedan Ctl. Där finns också villkorsinstruktionen if .. else.

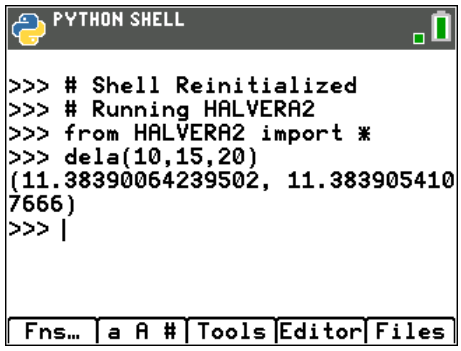
Tryck nu på f4 (Run) för att köra programmet. Trycka på vars och välj dela() och tryck sedan på Ok. Fyll nu i argumenten (10,15, 0.001). Tryck sedan på enter.

Nu får vi beräknade värden på ett smalt intervall. Med två korrekta decimaler blir svaret 11.38.

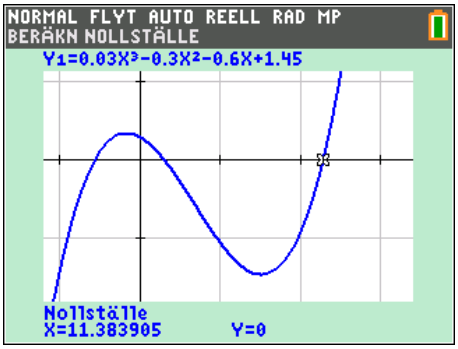
En variant som gör samma sak

Istället för att ge precision kan vi arbeta med ett antal *steg*. Istället för en while-sats så har vi nu en *for i in range(step)*-sats. Vid körning så skriver man t ex dela(10,15,20). Se resultat i skärmbilden till höger.





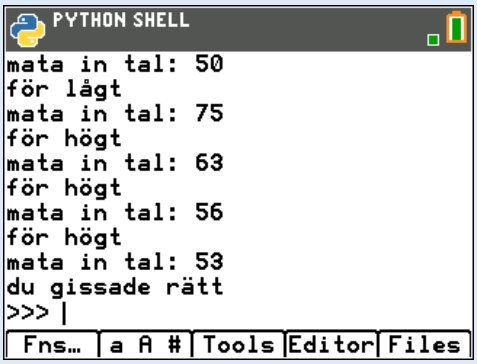
Med räknarens inbyggda verktyg för bestämning av nollställe får vi detta resultat. Man plottar då grafen och trycker sedan på [2nd][calc]. Välj sedan alternativ 2. Vi får resultatet 11,383905

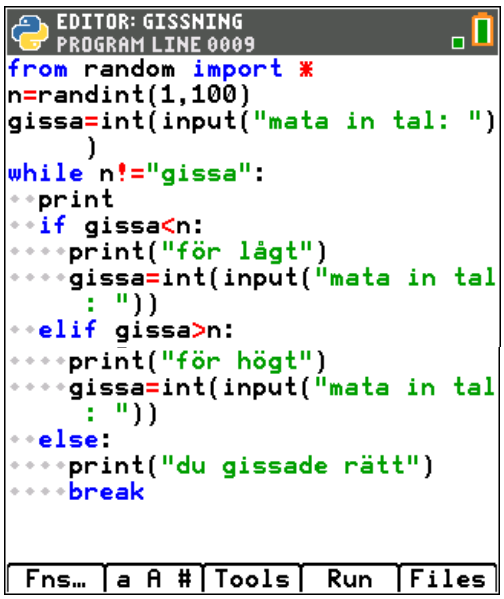


Lärarkommentar: På första sidan i denna övning så visade vi på strategin för att hitta en sida i en bok på 1024 sidor. Här har vi ett skript som fungerar på samma sätt. Användaren får alltså själv göra intervallhalveringar uppåt eller nedåt. Här har vi 100 som övre gräns men användaren kan ju själv bestämma denna gräns.

Vi har fuskat lite och satt ihop två skärmar på räknaren för att kunna visa hela programmet.

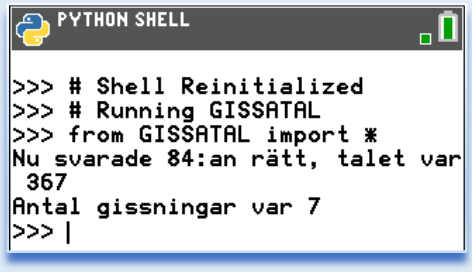
Här har vi en körning. Det räckte med 5 gissningar här.

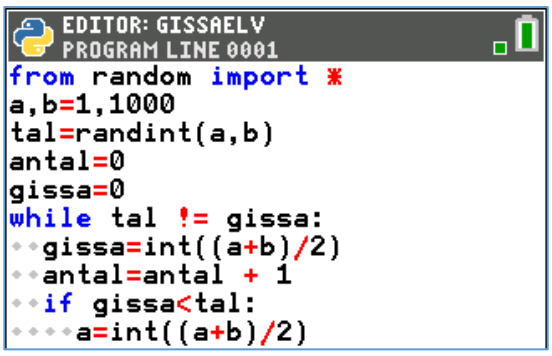




Visa detta skript för eleverna och gör några körningar. För att visa principen kan man börja med ett fåtal tal.

Den uppgift eleverna får här är att skriva klart skriptet. Vid körning ska det se ut som det visas på skärmen nedan.





Vi visar här bara den översta delen av skriptet. Eleverna uppgift är alltså att skriva klart! När de är klara och har provkört så ska de göra ett antal körningar och redovisa statistik på antal gissningar.