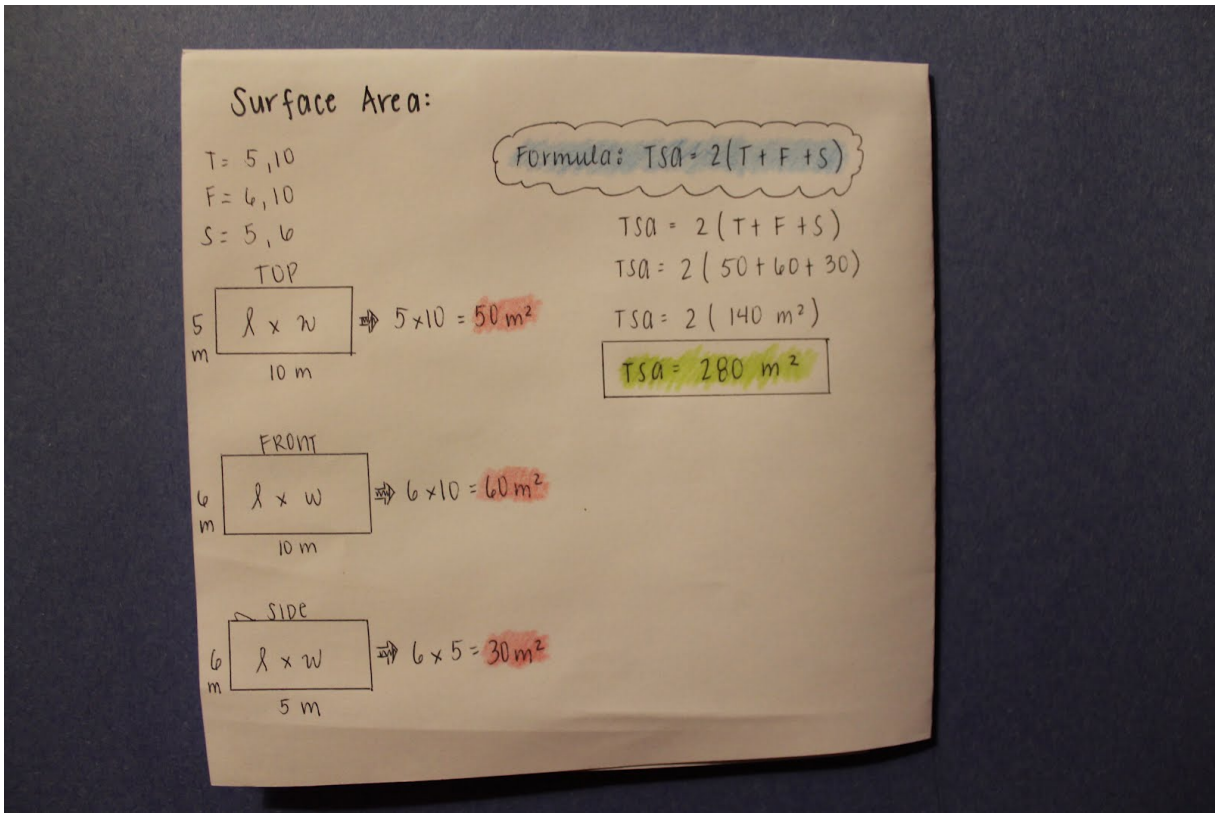
Ola Sahlin
Sannarpsgymnasiet
Frennarpsvägen
Halmstad
ola.sahlin@halmstad.se
070/7146327
Handledare: Veronica Gaspes

Didaktisk programmeringsuppgift

Delmoment inom kurs i programmering för lärare.

Ht 2018.



En introduktion i programmering med infärgning av matematik.

Innehållsförteckning

En introduktion i programmering med infärgning av matematik.

Bakgrund:
Vårt samhälle utvecklas i en riktning där samhällsmedborgaren möter behov av att besitta digitala färdigheter såväl privat som i olika yrkesroller. Skolverket har därför fått i uppdrag att utveckla läroplanen för grundskola och gymnasieskola i en riktning som stärker samhällsmedborgaren i ovan nämnda situationer.
Digitalisering av ämnet matematik är obligatoriskt inom skolan(såväl grundskola som gymnasieskola) from ht 2018. Digitalisering ska vara en naturlig del inom ämnet matematik och bidra till ökad förmåga att lösa matematiska problem av olika slag. Digitalisering inom matematik omfattar användandet av ett brett utbud av digitala verktyg såsom funktionsräknare, grafräknare, CAS-verktyg, kalkylark, geometriska verktyg mm samt programmering. Programmering utgör alltså en del av digitaliseringen av ämnet matematik.
För att öka kompetensen hos matematiklärarna har Skolverket uppdragit Sveriges högskolor att erbjuda fortbildning för matematiklärare.
Denna uppgift är en deluppgift i fortbildningskursen programmering för lärare om 7,5 p och har en didaktisk inriktning. Uppgiften knyter samman programmering och matematik.
Kursen genomförs av HH(Högskolan Halmstad). Ansvarig examinator är Veronica Gaspes.

Syfte:
Uppgiften syftar till:
- öka kunskaperna hos undervisande matematiklärare i olika programmeringsspråk, sätt att tänka inom programmeringsstrukturer.
- bereda undervisande matematiklärare undervisningsmateriel inom programmering.
- materielet syftar till att fördjupa kunskap, förmågor samt insikter i matematik genom att använda programmering som ett verktyg för att visualisera förlopp och underlätta beräkningar.

Teori:
Programmering handlar om att instruera en hårdvara(HW) eller del av en HW så att HW utför ett visst arbete. Exempel på HW är mikrodator, dator, robot, mobiltelefon mm.
Personen som utför detta hantverk brukar ofta kallas programmerare.
Programmerarens samlade instruktioner för en viss uppgift kallas program och instruktionerna skrivs i något programmeringsspråk. Programmering kan ske på olika nivåer i förhållande till HW:
-maskinkod: närmast HW.
- assemblerprogrammering
- allmänna programspråk: som behöver kompileras dvs översättas till maskinkod. En sats eller instruktion leder vanligvis till flera instruktioner på maskinkodsnivå.

Förenklat delas allmänna programmeringsspråk in i
- imperativa programmeringsspråk: sekvens av satser som ofta itereras. Ex Python, Pascal mfl
- deklarativa programmeringsspråk: funktionell och logikprogrammering där programmen är uppbyggda av deklarationer av funktioner definierade i termer av varandra och ofta rekursiva. Ex LISP, Scheme mfl.

Programmeringsspråk under denna kurs.
Kursen innehåller orientering i dels Scheme (deklarativt programmeringsspråk) och dels Python (imperativt programmeringsspråk).

Målgrupp.

Läromedlet nedan vänder sig till elever på gymnasiets högskoleförberedande program. Eleverna är då i åldrarna 15-19 år och i huvudsak mot de yngre åldrarna 15-16 år (åk 1 ma 1b eller ma 1c). I nuläget har eleverna inte med sig några egentliga kunskaper eller erfarenheter kring programmering. På sikt med de nya läroplanerna kommer detta succesivt att utvecklas mot elever med större erfarenhet av programmering.

Didaktiska övervägande och reflektioner.

Utformningen av läromedlet har sin grund i att eleven i stor utsträckning själv ska kunna ta till sig innehållet samt lösa uppgifter/övningar. Innehållet är tänkt att ge en grund i programmering med en tydlig infärgning av matematik. Lärarens roll grundar sig i bollplankets och stödjande funktion. Arbete med läromedlet sker parallellt med att man arbetar med programmeringskoden. Lärarens kan(har möjlighet att) träda in i lärprocessen på ett mer centralt ”katederbaserad” sätt under delar av lektionen/lektionerna. Läromedlet kan genom detta tidsanpassas från 3-4 lektioner upp till kanske 6-7 lektioner.
Eftersom utgångspunkten är att eleven ska skapa programmeringskod och lösningar på problem i så stor utsträckning finns det säkert delar som rent programmeringstekniskt skulle kunna utformas elegantare.
Läromedlet kan byggas på med fler moduler vilket möjliggör fördjupningar.

Läromedel

Välkommen till denna orienteringskurs i programmering.

Materiel:

Den som följer denna kurs behöver:

* penna,suddgummi, anteckningspapper.

* dator/chromebook med Dr Racket nedladdad alternativt plattformen Wescheme för att köra programspråket online.

* projektor.

* manual med schemes programkod i syntaxutförande:
  https://www.wescheme.org/doc/wescheme.html

Läromedlets uppbyggnad och struktur.
Kurs omfattar 8 moduler och syftar till att ge en grundläggande orientering i programspråket Scheme.

Modul 1: Plattformen Weschemes uppbyggnad, kodarea och interaktionsarea .

Modul 2: Grundläggande regler och begrepp för språket Scheme, syntax, arbetsytan Wescheme uppbyggnad mm

Modul 3: Några geometriska verktyg.

Modul 4: Att skriva text och att arbeta med strängar.

Modul 5: Hur man definierar och använder konstanter.

Modul 6: Hur man definierar funktioner.

Modul 7: problem: optimering av geometrisk rektangulär form med given stängsellängd(omkrets)

Modul 8: hur man skapar ett tomt fönster och lägger in geometriska figurer i detta. ***Ej klar.***

Fördjupning

Se Bootstrap algebra som vi läst.

Du som ***vill fördjupa*** dig vidare och tillverka ditt eget spel av typen Ninja cat hänvisas till
https://www.bootstrapworld.org/materials/spring2018/courses/algebra/english/
Du behöver då också en studiebok som du kan hämta på följande länk

https://www.bootstrapworld.org/materials/spring2018/courses/algebra/english/resources/workbook/StudentWorkbook.pdf

Modul 1.
Både Dr Racket och onlineplattformen Wescheme har 2 olika ytor eller eller område(areor):
a: kodområde/kodarea där varje rad är numrerat. Här skriver man programkod som sedan ska bilda ett program. När man trycker på enter byter man rad.
b: interaktionsområde/interaktionsarea där man kan anropa olika delar av programmet eller skriva in enstaka operationer mm. När man trycker på enter exekveras(utförs) operationen man skrivit in.
Se screencast nr 1: PP=PowerPoint.

Modul 2.
Samtliga ämnen och discipliner har sitt språk för att beskriva sin egne värld och verklighet. Detta gäller även för programmering. Här beskriver vi de grundläggande begreppen inom programmering.

Ibland kan sådant ”fikonspråk” upplevas som att man krånglar till situationen men syftet är raka motsatsen: att skapa ett beskrivande verktyg med hög skärpa som inte kan misstolkas. Inom programmering styr man en hårdvara som inte har någon erfarenhet att falla tillbaka på och som inte kan något mer än att utföra det som den blir tillsagd att utföra. Programmeringsspråket måste därför vara entydigt vilket betyder att det måste vara uppbyggt så att det inte kan misstolkas eller tolkas på flera olika sätt.

Det första begreppen som vi introducerar är

Operator: är inom matematiken en symbol som talar om vad man ska göra(representerar en matematisk operation). Exempel plus: + : addition, minus: - :subtraktion

Operander: De objekt som operatorn opererar(verkar på) på kallas för operander.

Input: det som matas in i en operation eller åtgärd i porgramkoden.

Output: det som kommer ut ur en operation eller åtgärd i en programkod.

Syntax: är uppsättningen regler som talar om hur kombinationer av symboler ska skrivas på ett korrekt sätt inom programmeringsspråket. Det är extremt viktigt att hålla koll på syntaxen i programmeringsspråket. I Scheme gäller några grundläggande basregler
* **operatorn**(”räknesättet”, det som ska göras skrivs först).
* man har alltid **mellanslag** mellan operator och operander. Viktigt.

Exempel och övning 1:

Skrivet språk	Syntax med matematiska symboler	Syntax Scheme
Fyra plus tjugotre	4 + 23	(+ 4 23)
Två gånger minus 3	2 * (-3)	(* 2 -3)
Fyra plus två gånger minus 3	4 + 2 * (-3)	(+ 4 (* 2 -3))
Femton minus tolv delat med 3		
	5 * 3 – 4 * 6	
		(/ (+ 11 9) (- 9 4)
	(14 + 6) * (11 – 9)	

a: Fyll i tabellen. Beräkna uttryckens värde manuellt(penna och papper).
b: Växla mellan arbetsboken och Wescheme och skriv in Schemesyntaxen för uttrycken i interationsytan och och tryck på enter(exekvera beräkningen i Wescheme):
c: Kontrollera och jämför.
d: Experimentera med olika uttryck i Scheme

Några inbyggda funktioner(operatorer) i Scheme:
I Scheme finns ett stort antal färdiga funktioner med en syntax som avviker något från hur vi skriver dessa i vanliga fall Nedan följer några exempel

Exempel och övning 2:

Skrivet språk	Syntax med matematiska symboler	Syntax Scheme
Två upphöjt med tre	2^3	(expt 2 3)
Tre i kvadrat	3^2	(sqr 3)
pi upphöjt med 2	π^2	(expt pi 2)
Kvadratroten ur fyra	$\sqrt{4}$	(sqrt 4)
----------	$3^2 + 2^3$	
----------		(sqrt (+ (sqr 3) (sqr 4)))
	$\sqrt{\pi^2}$	

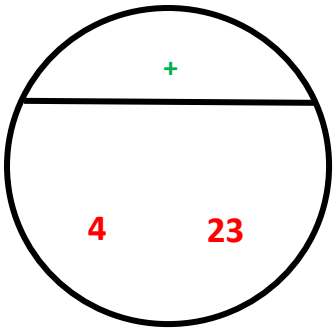
Verktyg för att definiera och dokumentera i samband med programmering.
När man ska skriva större program som innehåller flera olika delar som ska samarbeta med varandra är det viktigt att i detalj kunna beskriva vad de olika delarna i programmet ska göra vad som ska matas in och vad som ska skrivas ut. Vi introducerar här två verktyg som kan användas för att dokumentera dels på papper och dels inne i programkoden för att underlätta för läsaren av programkod(underhåll av programkod utveckling av programkoden).

a: **kravspecifikation input/output:** dokumentation av vilken dataform som förväntas i input resp vilken data som förväntas i outputen. Gör man inte detta noggrant är detta något som ofta ligger till grund för programmeringsfel(logikfel syntaxfel mm).

b: ”**Cirkel of evaluation”:** grafisk beskrivning av en operationsstruktur. Även detta utgör en god grund för att skriva programkod korrekt och med rätt logisk struktur.

Gå tillbaka till exempel och övning 1:
Första raden i den tabellen har skrivs matematiskt resp på Schemekod
matematiskt **4 + 23**
schemesyntax: (**+ 4 23**)
Ska man dokumentera detta **input/output** blir detta:
number number -> number
dvs två värden ger ett värde.

Ska man dokumentera detta i en ”**Cirkel of evaluation”** ser det ut så här:

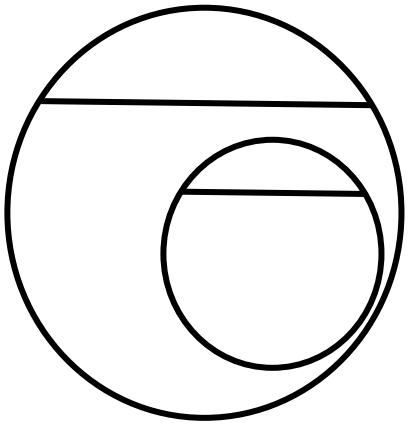


Operatorn(+) placeras ovanför strecket i cirkeln och **operanderna**(**4 och 23**) nedanför och i rätt turorning.

Tredje raden i den tabellen har skrivs matematiskt resp på Schemekod

4 + 2 * (-3)
(**+ 4** (*** 2 -3**))

”**Cirkel of evaluation”** ser det ut så här:



-

Den inre cirkeln utgör den inre schemeparentesen dvs (*** 2 -3**). Fyll själv i med penna.

Övning 3. Rita ”**Cirkel of evaluations ”** för de tre sista raderna i tabellen i exempel och övning 1.

Modul 3: Några geometriska operatorer samt operatorer som verkar på bilder.
Scheme innehåller gott om figuroperatorer:

a: operatorer som **genererar** geometriska figurer.

b: operatorer som **verkar på** geometrisk figurer eller bilder:

c: operatorer som **sätter samman** figurer/bilder till större enheter.

d: operatorer som verkar på figurer/bilder för att **utvärdera en egenskap** som figuren/bilden har.

I vår introduktion exemplifierar vi endast med några exempel. Vill man fördjupa sig hänvisar vi till kodkatalogen eller manualen ovan(under läromedel och materiel).

a: Nedan ger vi några exempel på operatorer som **genererar geometriska figurer**.

Ex: (rectangle 40 20 "outline" "red"):
(rectangle 20 40 "solid" "blue")

Kopiera koden via ctrl c och klistra in den i interaktionsarean i Wescheme och testa vad koden genererar för figur samt vilka
dimensioner figuren får:
vilken inverkan har
”outline”/”solid”:
”red”/”blue”

har för påverkan på utseendet?

Här pratar vi alltså om att operatorn rectangle som opererar på 2 taloperander och 2 strängoperander.
kravspecifikation input/output: number, number, string, string -> image
syntax: (rectangle number number string string)

Det går på snarlikt sätt att konstruera cirklar med cirkeloperatorn:
syntax: (circle number string string)
Exempel: rita en grön(green) solid cirkel med diametern 40 i interaktionsarean.

Ytterligare några exempel att testa:
(radial-star 8 8 64 "solid" "darkslategray")
Beskriv figuren som ritas upp:

Övning: Skriv om syntaxen så att en 6 spetsig figur ritas som är konturritad och blå och dessutom har kortare spetsar.

I manualen finns ett stort antal geometriska figurer beskrivna. Botanisera gärna.

b: operator som **verkar på** geometrisk figurer eller bilder:

Vi introducerar scale samt rotate-operatorerna:
Exempel scale:

```
(radial-star 6 8 64 "solid" "darkslategray")
```

```
(scale 2 (radial-star 6 8 64 "solid" "darkslategray") )
```

kopiera koden och lägg in den antingen i kodarean och kör programmet(RUN) eller lägg in koden i interaktionsarean och tryck enter.

Förklara vad scaleverktyget gör:

Hur gör man om man vill förminska en figur ? Testa genom att ändra koden.

Exempel rotate operatorn

```
(rotate 45 (rectangle 20 40 "solid" "red"))
```

kopiera koden och lägg in den antingel i kodarean och kör programmet(RUN) eller lägg in koden i interaktionsarean och tryck enter.

Förklara vad rotateverktyget gör:

Övning: Betrakta följande bild : (bitmap/url "http://bootstrapworld.org/images/icon.gif")

rotera följande bild 180 grader.

Är dessa båda bilder varandras spegelbilder om man försöker göra en horisontell respektive vertikal spegling?

c: operatorer som **sätter samman** figurer/bilder till större bilder(enheter).

Vi tar upp två operatorer (men det finns många fler) **above** och **beside**:
- above-operatorn: lägger 2 eller flera bilder ovanför varandra:

Betrakta figuren (square 20 "solid" "red"). Vi vill bygga samman flera av denna figurtyp ovanför varandra.

syntax: (above (bild1) (bild2) (bild3)) Detta kan se ut så här:

(above (square 40 "solid" "blue") (square 30 "solid" "green") (square 20 "solid" "red"))

Testa i Wescheme vad du får ut. (kopiera koden).

Övning above: Gör ett rödljus med hjälp av above.

- beside-operatorn: lägger flera 2 eller flera bilder vid sidan om varandra:

syntax: (beside (bild1) (bild2) (bild3)) Detta kan se ut så här:

(beside (square 40 "solid" "blue") (square 30 "solid" "green") (square 20 "solid" "red"))

Testa i Wescheme vad du får ut. (kopiera koden).

Övning beside: lägg de tre kvadraterna i storleksordning från den lilla längst till vänster och den stora längst till höger med hjälp av above.

Övning above och beside: gör ett luffarschacksunderlag som har färgerna svart och vitt.

d: operatorer som verkar på figurer/bilder för att **utvärdera en egenskap** som figuren/bilden har.

Vi tar upp två operatorer (men det finns många fler) **image-width** och **image-height**:

- image-width-operatorn:

syntax: (image-width (bild)) ,

Detta kan se ut så här:

(image-width (circle 20 "solid" "green"))

Vilket resultat får du när du kör koden ? Förklara vad som anges.

Ange dessutom **kravspecifikation input/output:** ????-> ??

- image-height-operatorn:

syntax: (image-height (bild))

Detta kan se ut så här:

(image-height (rectangle 20 50 "solid" "green"))

Vilket resultat får du när du kör koden ? Förklara vad som anges.

Ange dessutom **kravspecifikation input/output:** ????-> ??

Fördjupning: Undersöka operatorn place-image/align:

Skriv in följande kod:

```
(place-image/align (triangle 48 "solid" "yellowgreen")
                   64 64 "right" "bottom"
                   (rectangle 64 64 "solid" "mediumgoldenrod"))
```

a: Förklara vad verktyget gör, kravspecifikation input/output .
b: Skriv om koden på två olika sätt så att den liksidiga triangeln hamnar längst ner till vänster.

Modul 4: Att skriva text och att behandla text i scheme.
I scheme finns det goda möjligheter att
- skriva text
- modifiera textens storlek, färg mm
- dela upp text, plocka ut olika delar av en text, förstabokstaven, sista bokstaven, första ordet
- sätta samman text

Ex 1. Kopiera **koden nedan** och lägg in i interaktionsarean samt tryck på enter.

(text "Sannarpsgymnasiet är en bra gymnasieskola" 18 "black")

Som du ser skrivs texten **Sannarpsgymnasiet är en bra gymnasieskola ut som output.**
Texten här kallas för en sträng. Vill man få ut en text som utskrift använder man operatorn **text**.
Övning 1. Modifiera koden så att texten blir större. Ge exempel på hu koden då kan se ut.

Svar:

Övning 2. Modifiera koden så att stängen skrivs ut med grön text istället.

Svar:

Övning 3. Utvärdera vad operatorn string-append gör genom att kopiera följande kod till interaktionsarean

(string-append "Kalle" "kula")

Förklara vad operatorn string-append gör här:

Övning 4. Du har två olika textsträngar:
1. Halmstads BK
2. spelar i blå tröjor

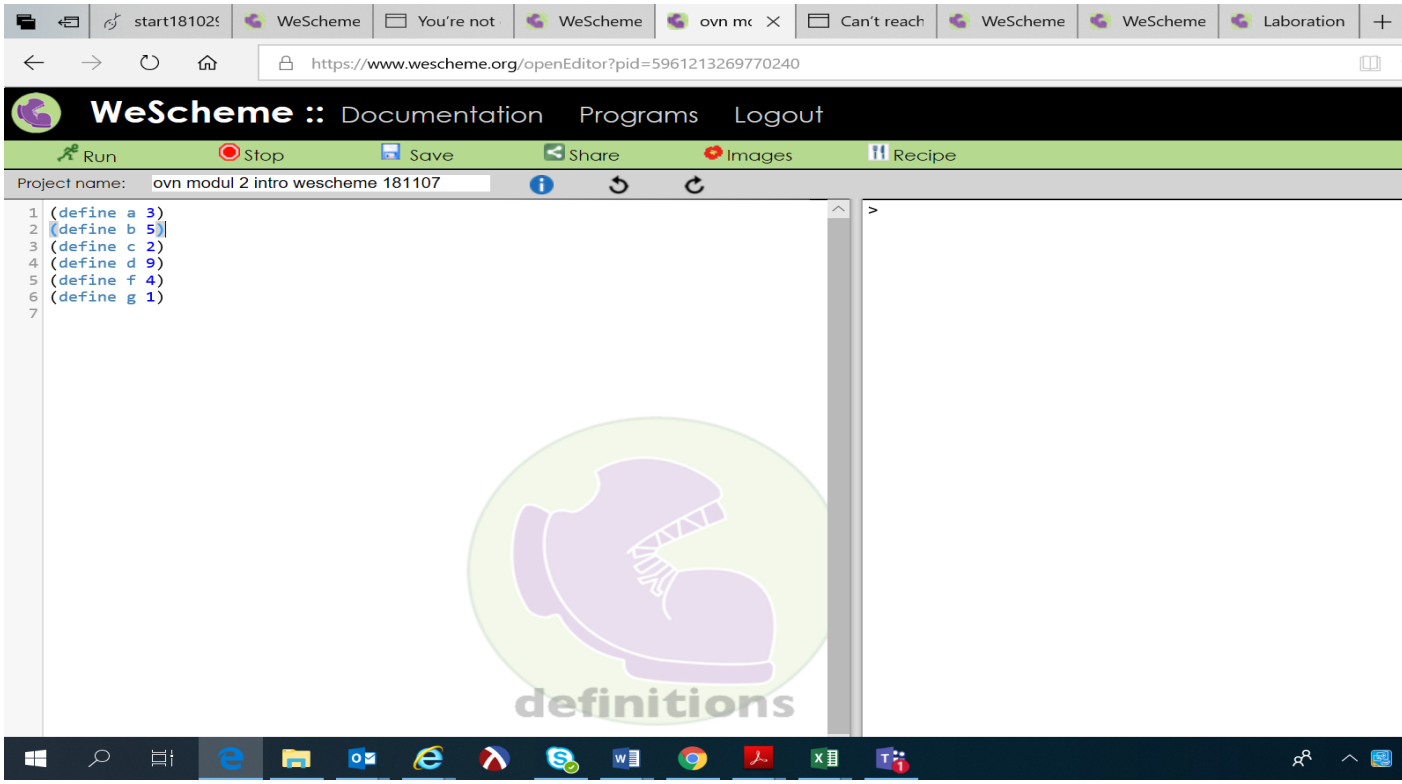
Skriv en kod som sätter samman dessa 2 textsträngar till textsträngen

Halmstads BK spelar i blå tröjor

Svar: Skriv koden här:

Det finns fler operatorer som arbetar medsträngar, se länken med till manual med programkod ovan.

Modul 5: Hur man definierar och använder konstanter.
Inom programmering är det vanligt att man definierar konstanter vilket betyder att man tilldelar bokstäver (vanligtvis) olika värden. Dessa blir sedan viktiga verktyg i programstrukturen. I scheme kan det se ut som i bilden nedan.



Definierar konstanterna i Weschemes programkodsyta så som bilden visar.
övning 1.Vad kommer följande kod att ge för värde?

(+ a b)

(+ g (* a c))

Förklara varför koden ger de värden som svar

övning 2.

a: Vad kommer följande kod att ge för värde ?

(/ (+ a b c d f g) 6)

b: Skriv om programkoden i övning 2a till ett matematiskt uttryck.

b: Vad kalls det du räknat ut med koden i övning 2a ?

övning 3

Skriv in följande kod i wescheme i programkodsytan under definitionerna av konstanterna.

(< a b) Vad får du för svar ? Förklara.

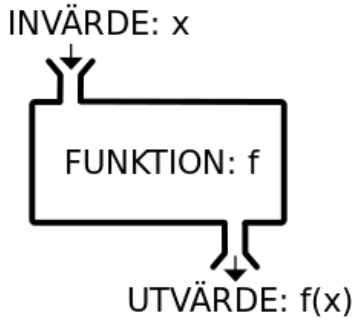
(> (expt c a) d) Vad får du för svar ? Förklara.

Modul6: Hur man definierar och använder funktioner.

Inom matematiken och en del andra vetenskaper är begreppet funktion ett viktigt begrepp och verktyg. En funktion kan liknas vid en maskin som tar in ett ” invärde” och ger ett svar utvärde. Funktioner har två viktiga egenskaper:

a: Funktionen ger varje gång man matar in ett och samma invärde samma utvärde.

b: För varje invärde ger funktionen alltid bara ett utvärde (svar).



Ett viktigt begrepp är

variabel=något som kan varieras. Inom matematik kan variabeln anta olika värden.

Inom programmering är funktionsbegreppet också viktigt.

Syntax: (define (funktionsnamnet variabel/variabler) (funktionskroppen))

Funktionskroppen kan vara ett matematiskt uttryck eller liknande men kan också vara geometriska verktyg(se modul 3 ovan).

Nedan följer några exempel skrivet i schemekod:

Exempel 1:
(define (**Kostnad x**) **(+ x 15)**)

Kostnad: namnet på funktionen
x: variabeln är x och som kan varieras i värde
(+ x 15): funktionskroppen som beräknar Kostnad genom att addera(+) värdet på x med 15.

Vad kommer sedan följande kod att ge för svar ? Prova gärna i Wescheme.

(Kostnad 5)

(Kostnad 20)

Exempel 2:
(define (**Cirkelbild r**) **(circle r ”solid” ”red”)**)

Cirkelbild: namnet på funktionen
r: variabeln är r och som kan varieras i värde och beskriver radien på cirkeln.
(circle r ”solid” ”red”) är funktionskroppen och ritar en kompakt röd cirkel med den radie som som man önskar då då man anger värdet på r.

Vad kommer sedan följande kod att ge för svar ? Prova gärna i Wescheme.

(Cirkelbild 20)

(Cirkelbild 40)

Exempel 3:
Följande kod i wescheme beskriver en funktion som beräknar kostnaden att hyra en vindsurfingbräda.

(define (kostnad tid) (+ (* 100 tid) 50))
a: Hur mycket kostar det att hyra vindsurfingbrädan 3 h?
b: Hur stor är den fasta respektive rörliga kostnaden ?

Arbetsgång då man definierar funktioner

I programmering är det viktigt att ha kontroll över vilken typ av indata funktionen kräver samt vad funktionen producerar för resultat eller utdata. Det är annars lätt att man som indata ger funktionen fel typ av data vilket resulterar i att funktionen inte fungerar eller ger fel utdata. När man sedan ska bygga samman flera funktioner i större strukturer underlättar en god dokumentation av respektive funktion möjligheterna till ett gott resultat.

Denna arbetsgång kan delas in i 3 olika faser eller delar:

A: syfte och funktionskontraktsfas: i detta kontrakt anger man 3 saker
- funktionen tilldelas ett namn: gärna kopplat till vad funktionen gör
- man talar om och beskriver funktionens indata(domain)
- samt talar och beskriver funktionens utdata(range)

Sammansatt på en rad blir det:

_____: _____ -> _____
funktionens namn domain range

B: Exempel och mönsterfasen:

När man själv ska definiera funktioner är det viktigt att se det mönster för hur en beräkning ska göras. Mönstret kan sedan generaliseras med ett (ibland algebraiskt) uttryck som i scheme bildar själva funktionskroppen. Det kan då vara till stor hjälp att sätta upp några(2-3 eller så många så att mönstret blir tydligt) syntaxexempel under varandra.

Det brukar då vara lättare att se själva mönstret som sedan bildar funktionskroppen i ett algebraiskt uttryck.

Här skriver man exemplen enligt följande struktur som påminner om syntaxen för hur funktionen sedan ska definieras i scheme:

exempel1:(_____) (_____)
funktionens namn ex på inputs(indata) det som funktionen sedan ska beräkna eller producera)

exempel2:(_____) (_____)
funktionens namn ex på inputs(indata) det som funktionen sedan ska beräkna eller producera)

exempel3:(_____) (_____)
funktionens namn ex på inputs(indata) det som funktionen sedan ska beräkna eller producera)

Ringa sedan in (vertikalt under varandra) de delar som sedan ändras. Dessa inringade delar ska du sedan tilldela variabelnamn i din slutliga funktionsdefinition, fas 3 se nedan.

C: Definitionsfas:

I denna fas skriver man upp(definierar) den slutliga funktionen med precis schemesyntax med variabler och funktionskropp enligt

(define (_____) (_____))
funktionens namn variabel/variabler funktionskroppen=det som funktionens ska göra med variabeln/variablerna

Nedan följer några exempel på hur en sådan arbetsgång kan göras

Exempel 1: En löpare springer med hastigheten 3 meter per sekund. Skriv en funktion som kallas löparsträcka och som ska använda tiden som löparen sprungit i sekunder sedan start och anger(producerar) hur långt löparen har hunnit efter start i meter.

Arbetsgången kan då se ut enligt följande:

A: **syfte och funktionskontraktsfas:**

_**löparsträcka**_____: __värde ____ -> ___värde _____
funktionens namn domain range

Förklaring: Funktionen heter **löparsträcka** och behöver ett värde(den tid som löparen sprungit) för att kunna räkna ut ett annat värde: hur långt löparen sprungit.

B: Exempel och mönsterfasen:

exempel1:(**löparsträcka** **2**) (* **3** **2**)
funktionens namn ex på inputs(indata) det som funktionen sedan ska beräkna eller producera)

exempel2:(**löparsträcka** **5**) (* **3** **5**)
funktionens namn ex på inputs(indata) det som funktionen sedan ska beräkna eller producera)

exempel3:(**löparsträcka** **7**) (*** 3 7**)
funktionens namn **ex på inputs(indata)** **det som funktionen sedan ska beräkna eller producera)**

Förklaring: Ringa vertikalt in det som ändras. Det som man ändrat i de olika exemplen är **löpartiden.** Funktionen **löparsträcka** behöver en variabel som man skulle kunna kalla **tid och som anger värdet av den tid löparen sprungit. För att beräkna (producera) värdet av den sträcka som löparen sprungit multiplicerar man löparens hastighet med den tid som löparen sprungit.**

C: Definitionsfas:
Här definierar man den slutliga funktionen med define som skulle kunna se ut enligt följande:

(define (**löparsträcka** **tid**) (*** 3 tid**))
funktionens namn variabel/variabler funktionskroppen=det som funktionens ska göra med variabeln/variablerna

Exempel 2: Skriv en funktion som kallas rek och som ska använda ett inmatat värde som bredd för en rektangel. Funktionen ska producera en bild av en massiv orange rektangel där höjden är dubbelt så stor som den inmatade angivna bredden på rektangeln.

A: **syfte och funktionskontraktsfas:**

_**rek**_____: __värde ____ -> ___bild _____
funktionens namn domain range

Förklaring: Funktionen heter **rek** och behöver ett värde(rektangelns bredd) för att kunna producera(rita ut) en bild på en massiv orange rektangel där höjden är dubbelt så stor som bredden.

B: Exempel och mönsterfasen:

exempel1:(**rek** **20**) **(rectangle 20 (* 2 20) ”solid” ”orange”)**
funktionens namn ex på inputs(indata) det som funktionen sedan ska beräkna eller producera)

exempel2:(**rek** **30**) **(rectangle 30 (* 2 30) ”solid” ”orange”)**
funktionens namn ex på inputs(indata) det som funktionen sedan ska beräkna eller producera)

exempel3:(**rek** **50**) **(rectangle 50 (* 2 50) ”solid” ”orange”)**
funktionens namn ex på inputs(indata) det som funktionen sedan ska beräkna eller producera)

Förklaring: Ringa vertikalt in det som ändras. Det som man ändrat i de olika exemplen är **bredden.** Funktionen **rek** behöver en variabel som man skulle kunna kalla **bredd och som anger värdet av rektangelns bredd. För att rita (producera) rektangelns behövs verktyget rectangle, se modul 3 a verktyg som producerar geometriska figurer. Verktyget rectangle behöver 4 argument, bredd, höjd, typ av rektangel(ifylld eller kontur) samt färg. Höjden beräknas genom att multiplicera bredden med 2.**

C: Definitionsfas:
Här definierar man den slutliga funktionen med define som skulle kunna se ut enligt följande:

(define (**rek** **bredd**) **(rectangle bredd (* 2 bredd) ”solid” ”orange”)**)
funktionens namn variabel/variabler funktionskroppen=det som funktionens ska göra med variabeln/variablerna

Övning 1.
a: Definiera en funktion i wescheme som heter Taxikostnad och som beräknar kostnaden för en taxitur som är x mil med följande förutsättningar:
startavgift(framkörningsavgift)= 50 kr samt körkostnaden 250 kr per mil.
OBS! Gå igenom de 3 stegen eller faserna A, B resp C ovan i arbetsgången.
A: syfte och funktionskontraktsfas:

_____: _____ -> _____
funktionens namn domain range

B: Exempel och mönsterfasen:

exempel1:(_____) (_____)
funktionens namn ex på inputs(indata) det som funktionen sedan ska beräkna eller producera)

exempel2:(_____) (_____)
funktionens namn ex på inputs(indata) det som funktionen sedan ska beräkna eller producera)

exempel3:(_____) (_____)
funktionens namn ex på inputs(indata) det som funktionen sedan ska beräkna eller producera)

C: Definitionsfas:

(define (_____) (_____))
funktionens namn variabel/variabler funktionskroppen=det som funktionens ska göra med variabeln/variablerna

b: Skriv in funktionen i kodområdet/kodarean, kör run.
Testa funktionen genom att beräkna kostnaden för en Taxiresa på 7 mil.

Övning 2
a: Definiera en funktion i wescheme som heter traktorhjul. Funktionen ska producera en bild med 2 st olika stora svart cirkulära hjul bredvid varandra där det större bakhjulet är 50 % större än det mindre framhjulet. De båda hjulens storlek ska bara bero av samma variabel som du kan döpa till tex radie. Avståndet mellan hjulens ytterkanter ska vara lika stort som diametern på framhjulet. Enbart det stora hjulet ska ”nå marken”. OBS! Gå igenom de 3 stegen eller faserna A, B resp C ovan i arbetsgången.

A: syfte och funktionskontraktsfas:

_____: _____ -> _____
funktionens namn domain range

B: Exempel och mönsterfasen:

exempel1:(_____) (_____)
funktionens namn ex på inputs(indata) det som funktionen sedan ska beräkna eller producera)

exempel2:(_____) (_____)
funktionens namn ex på inputs(indata) det som funktionen sedan ska beräkna eller producera)

exempel3:(_____) (_____)
funktionens namn ex på inputs(indata) det som funktionen sedan ska beräkna eller producera)

C: Definitionsfas:

(define (_____) (_____))
funktionens namn variabel/variabler funktionskroppen=det som funktionens ska göra med variabeln/variablerna

b: Testa funktionen genom att rita en traktorhjulen då det mindre hjulet har radien 20 enheter.

Modul7: Optimeringsmodul.

Beskrivning av slutproblem.

En rektangel har omkretsen 200 cm. Du ska skriva ett litet program(egentligen komplettera programstrukturen nedan) i wescheme som undersöker vilka mått rektangeln ska ha för att få så stor area som möjligt. Du ska kalla en av sidorna för s.
Programmet ska rita upp rektanglar och beräkna arean för varje för varje värde på s enligt tabellen nedan se delsteg 5:

Programstruktur

Nedan följer en programstruktur som du ska arbeta med och fylla i vissa delar, skriva beskrivande text, konstruera geometriska bilder, definiera funktioner mm. Du ska börja med att kopiera hela programstrukturen till weschemes kodarea genom att

1. **VARA NOGGRANN!**
2. Markera hela den feta texten till och med ;;;
3. Kopiera koden via ctrl c
4. Byt till weschemses kodarea och kopiera in koden via ctrl c.

; Programkodsstruktur för optimering av rektangelarea: Vilka mått ska en rektangel med omkretsen 200 cm ha för att arean ska bli så stor som möjligt ?

;resetfunktionen definieras

(define (reset s ke)

100)

; förutsättningar och frågeställning skrivs ut då programmet körs!

; Vad vill du att programmet skriver ut för text inledningsvis?

;DelA

; Konstruktion av funktions som ritar rektangelbild med verktyget rectangle:

; På följande 2 rader ska du definiera en funktion med namnet number->square som ska tillverka(rita)en grön rektangel där en sida är s och den andra sidan sådan att omkretsen totalt blir 200 cm, hur lång är den andra sidan då?

;del B

;Konstruktion av funktionen rektangelarea som beräknar arean av rektangeln som ritas ut

;På följande rad ska du definiera funktionen rektangelarea som beräknar arean där en sida är s

;del C

; funktionen space på raden under ; del D används enbart för att ge mellanrum mellan de olika rektanglarna som ritas ut vid sidan om varandra.

;del D

; funktionen rektmedarea används för att montera samman rektangelbild med areavärde så att rektangelbilden läggs under rektangelareavärdet genom att använda vertyget underlay. Areavärdet skrivs ut som en sträng och måste konverteras från numeriskt värde till sträng via number->string samt verktyget för text som är just text och som behöver 3 argument: sträng, storlek samt färg

```
(define (rektmedarea s) (underlay (frame (number->square s)) (text (number->string
(rektangelarea s)) 12 "black" ) ))
```

;del E fyll i de värden på s (som finns i tabellen under deluppg 5) där du hittar ett frågetecken(?).

; Här under ritas samtliga rektanglar ur tabellen ut vid sidan om varandra. För att göra detta används bildverktyget beside. Du ska mata in de s-värde som finns i tabellen på respektive ställe för funktionen rektmedarea totalt 14 st.

```
 (beside (rektmedarea ?) (rektmedarea ?) (space 2) (rektmedarea ?) (space 2) (rektmedarea ?)
(space 2) (rektmedarea ?) (space 2) (rektmedarea ?) (space 2) (rektmedarea ?) (space 2)
(rektmedarea ?) (space 2) (rektmedarea ?) (space 2) (rektmedarea ?) (space 2) (rektmedarea ?)
(space 2) (rektmedarea ?) (rektmedarea ?) (rektmedarea ?) )
```

; För att följa hur arean ändras mer stegvis då sidan s ändras använder vi en big-bangfunktion. Detaljerna hur en big-bangfunktion är uppbyggd ligger utanför denna kurs. Det som ritas ut är rektmedarea som ändras genom knapparna up(pil-upp) och down(pil-ner) steg för steg då man trycker ner dessa. Man måste börja med ´pil-upp annars havererar big-bang.

```
(define (stop? world)
  (> world 99))
(big-bang 0
          [to-draw rektmedarea]
          [stop-when stop?]
          [on-key handle-key ] )
(define (handle-key x k)
  (cond [ (key=? k "up") (add1 x)]
        [ (key=? k "down") (sub1 x) ]
        [ else x]))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

Utveckling av den programgrundstruktur som du nu kopierat till kodarean i wescheme..

Du ska nu stegvis(deluppgifter) utveckla den grundkod som du kopierat till wescheme.

Delsteg 1:
Letat upp
;del A
i programkoden.
Du sk nu lägga in programkod på de 2 raderna under **;del A** så att följande text skrivs ut då programmet körs(RUN):

Varje rektangel har omkretsen 200 cm

Värdet i rektanglarna är rektangelns area

Texten ska skrivas ut på 2 rader med storlek 18 och med svart text.

Delsteg 2:
Letat upp
;del B
i programkoden.

På följande 2 rader ska du definiera en funktion med namnet **number->square** som ska tillverka(rita)en grön rektangel där en sida är **s (=variabel)** och den andra sidan sådan att omkretsen totalt blir 200 cm, hur lång är den andra sidan då?
Gå igenom den 3 stegs arbetsgång som man då man definierar funktioner:

A: syfte och funktionskontraktsfas:

number->square _____: _____ -> _____
funktionens namn domain range

B: Exempel och mönsterfasen:

exempel1:(_____) (_____)
funktionens namn ex på inputs(indata) det som funktionen sedan ska beräkna eller producera)

exempel2:(_____) (_____)
funktionens namn ex på inputs(indata) det som funktionen sedan ska beräkna eller producera)

exempel3:(_____) (_____)
funktionens namn ex på inputs(indata) det som funktionen sedan ska beräkna eller producera)

C: Definitionsfas:

(define (_____) (_____))
funktionens namn variabel/variabler funktionskroppen=det som funktionens ska göra med variabeln/variablerna

För in definitionen av number->square på raden eller raderna under **;del B**

Delsteg 3:
Letat upp
;del C
i programkoden.

På följande 2 rader i programkoden ska du definiera en funktion med namnet **rektangelarea** som ska **beräknar arean av den gröna rektangeln som ritas ut** (se del B med funktionen **number->square)** där en sida är **s (=variabel)** och den andra sidan sådan att omkretsen totalt blir 200 cm, hur lång är den andra sidan då?
Gå igenom den 3 stegs arbetsgång som man då man definierar funktioner:

A: syfte och funktionskontraktsfas:

rektangelarea _____: _____ -> _____
funktionens namn domain range

B: Exempel och mönsterfasen:

exempel1:(_____) (_____)
funktionens namn ex på inputs(indata) det som funktionen sedan ska beräkna eller producera)

exempel2:(_____) (_____)
funktionens namn ex på inputs(indata) det som funktionen sedan ska beräkna eller producera)

exempel3:(_____) (_____)
funktionens namn ex på inputs(indata) det som funktionen sedan ska beräkna eller producera)

C: Definitionsfas:

(define (_____) (_____))
funktionens namn variabel/variabler funktionskroppen=det som funktionens ska göra med variabeln/variablerna

För in definitionen av number->square på raden eller raderna under **;del C**

Delsteg 4:
Letat upp
;del D
i programkoden.

På följande rad i programkoden ska du definiera en funktion med namnet **space** som enbart används för att ge ett rektangulärt mellanrum mellan de olika **gröna rektanglarna** som ska ritas ut vid sidan om varandra. Du ska kunna variera storleken på utrymmet mellan de utritade gröna rektanglarna så space ska bero av en variabel som heter d.
Gå igenom den 3 stegs arbetsgång som man då man definierar funktioner:

A: syfte och funktionskontraktsfas:

space_____: _____ -> _____
funktionens namn domain range

B: Exempel och mönsterfasen:

exempel1:(_____) (_____)
funktionens namn ex på inputs(indata) det som funktionen sedan ska beräkna eller producera)

exempel2:(_____) (_____)
funktionens namn ex på inputs(indata) det som funktionen sedan ska beräkna eller producera)

exempel3:(_____) (_____)
funktionens namn ex på inputs(indata) det som funktionen sedan ska beräkna eller producera)

C: Definitionsfas:

(define (_____) (_____))
funktionens namn variabel/variabler funktionskroppen=det som funktionens ska göra med variabeln/variablerna

För in definitionen av number->square på raden eller raderna under **;del D**

Delsteg 5:
Letat upp
;del E
i programkoden.

Studera funktionen rektmedarea som är fördefinierad på en rad strax under **;del D** i programkoden. Funktionen rektmedarea används för att montera samman rektangelbild med areavärde så att rektangelbilden läggs under rektangelareavärdet genom att använda vertyget underlay. Areavärdet skrivs ut som en sträng och måste konverteras från numeriskt värde till sträng via number->string samt verktyget för text som är just text och som behöver 3 argument: sträng, storlek samt färg.

Nedan följer en delvis ifylld tabell. Den vänstra kolumnen innehåller de värden på sida s i de gröna rektanglarna som ska ritas ut. Du ska fylla i dessa värden som variabelvärden för funktionen rektmedarea. Fyll i värdena i den del där rektanglarna ritas ut vid sidan om varandra med hjälp av verktyget beside. Du hittar denna del under **;del E** i programkoden strax under **;del D.** Varje frågetecken (?) ska bytas ut mot ett värde på s i tabellen.

Efter att du gjort delsteg 5 klart ska du köra programkoden. Tryck på RUN. När du sedan kört programmet ska du fylla i tabellens högra kolumn med de areavärden som skrivs ut ovanpå respektive rektangel som ritas ut.

Sida rektangel i cm, s	Area rektangel fylls i sedan du kört programmet och läst av areavärdena i interaktionsarean
5	
7	
10	
15	
20	
30	
40	
50	
60	
70	
80	
90	
93	
95	

Delsteg 6: **Vilka slutsatser kan du dra av resultatet i tabellen ?**

a: Vilken är den största arean ? Svar:_____

b: Vilka mått har den rektangel som har störst area i tabellen ?
Svar:_____

c: Kan du se några symmetrier i de areavärden som finns i tabellen ? Beskriv dessa.

Har vi verkligen hittat den största arean ur de värden som finns i tabellen ?

I nästa delsteg(***delsteg 7***:) kommer programkoden då den körs(RUN) att stega fram rektanglar med mindre förändringar av sidorna än vad som vi gjort då vi konstruerat rektanglar med värden ur tabellen.
Du ska du använda tangenten ”pil upp” och trycka på pil upp succesivt. Då kommer rektangelns dimensioner(bredd och höjd) att ändras med kortare steg än vi gjort u tabellen. Samtidigt kommer arean att anges.

Delsteg 7:
Fortsätt att köra programmet genom att trycka pil upp. Du kan också backa med pil ner.

a: Hur ser rektangeln ut från början och vilken area har den då ?
Svar:

b: Beskriv hur rektangelns utseende och area ändras då du stegar med pil upp.

Svar:

c: Kommer du fram till samma resultat vad gäller största area, symmetrier och de mått som rektangeln har då som du fick ur tabellen ?

Svar:

Delsteg 8:
Här ska du använda din befintliga programkod som du utvecklat i delsteg 1-5 men modifiera programkoden på lämpligt sätt. Spara gärna en kopia på den programkod du gjort och ändra denna sedan.

Vi har undersökt hur arean ändras hos rektanglar som alla har omkretsen 200 ”cm.”
Undersök hur arean ändras hos rektanglar med några andra omkretser än just 200 cm.
Använd din befintliga programkod men ändra denna på lämpliga ställen så att du undersöker rektanglars area som har följande omkrets:
a: rektanglar med omkretsen 160 cm. Modifiera programkoden och kör sedan programmet med den modifierade koden. Vilken är den största arean du då får samt vilka mått har rektangeln då ?

b: rektanglar med omkretsen 120 cm. Modifiera programkoden och kör sedan programmet med den modifierade koden. Vilken är den största arean du då får samt vilka mått har rektangeln då ?

c: rektanglar med omkretsen 180 cm. Modifiera programkoden och kör sedan programmet med den modifierade koden. Vilken är den största arean du då får samt vilka mått har rektangeln då ?

d: Kan du se några mönster i de resultat du får som gör att du direkt kan dra slutsats vilka mått en rektangel med en given omkrets ska ha för att få så stor area som möjligt ?

Didaktisk sammanfattning.

Utgångspunkten med detta materiel i programmering har varit i så stor utsträckning som möjligt låta eleven på egen hand upptäcka delar av ett programkodsspråk. Via verktygen för att kopiera(ctrl c) och klistra in (ctrl v). Man kan då enkelt testa kod och modifiera denna och på så sätt ”upptäcka” kodens syntax och egenskaper i vad den gör.
Min erfarenhet som lärare säger mig att det krävs att man avsätter flera lektioner för att genomföra en sådan här uppgift med eleverna. Eleverna behöver tid för att smälta syntaxen i programkoden. Främst gäller detta beräkningskod där operatorn skrivs först och operanderna därefter vilket skiljer sig från det sätt som man behandlar matematik på ”traditionellt sätt”. Samtidigt bör man uppnå en fördjupad förståelse för hur matematiska operationer är uppbyggda och hur de fungerar.

Elevmaterielet ovan kan köras i en följd sammanhängande. Det är också möjligt att dela upp de olika modulerna vid olika tillfällen under den pågående kursen. Exempel på detta kan vara:
Modul 1 och 2 i samband med grundläggande aritmetik i kursen.
Modul 3 i samband med geometrin i kursen.
Modul 5 i samband med grundläggande algebra i kursen.
Modul 6 i samband med /eller efter introduktionen av funktioner i kursen.
Genom att dela upp elevmaterielet kopplat till olika delar inom kursen kan man få positiva effekter av att det teoretiska stoffet får sjunka in och ”sorteras” i elevens inlärningsprocess.
En risk med att dela upp materielet och lägga in det vid olika delar av kursen kan vara att eleverna löper risk att ”glömma” mellan de olika tillfällena man arbetar med materielet. Att glömma kräver eleverna förmår sig att ”repetera” för sig själv. Hur man till sist gör beror i stor utsträckning på hur den aktuella elevgruppens egenskaper, studieklimat, förmåga att själv äga sin lärprocess mm.

Utvecklingsmöjligheter programkoden.

Den sista delen i programkoden involverar en stegvis förändring av rektangelns utseende och area via en bigbang. Objektet som ändras dvs rektangeln skulle kunna placeras i en sk ”Background”. Vi har tidigare gjort snarlika sådan övningar. Dessa övningar har då byggt på att man har en och samma figur eller bild(med samma dimensioner och utseende) som placeras i Backgrounden eller förflyttas inom denna. Bildens utseende har hela tiden varit den samma. I min programkod ändras bildens utseende och dimensioner men skulle kunna placeras på samma ställe i ”Backgrounden”. Jag har försökt lösa detta programkodstekniskt utan att lyckas. Problemet har varit att jag får felmeddelande då jag använt mig av en funktion och inte en bild då jag placerat bilden i ”Backgrounden”:
Ex: (place-image DOT 50 50 BACKGROUND)) kräver att DOT är en bild och inte en funktion.
Jag har inte lyckats hantera detta själv programmeringstekniskt. Här skulle jag gärna få hjälp vilket skulle ge en elegantare slutdel då BigBangen körs.
Det skulle också vara intressant att plotta tabellen i delsteg 5 i en graf då programmet körs.

Man skulle också kunna ta uppgiften ett steg till genom att introducera en rektangel där ena sidan ”inte behöver ett stängsel” tex genom att rektangeln ligger mot ”en husvägg eller mot ett vattendrag”.

Facit programkod:

; Programkodsstruktur för optimering av rektangelarea: Vilka mått ska en rektangel med omkretsen 200 cm ha för att arean ska bli så stor som möjligt ?

;resetfunktionen definieras

(define (reset s ke)

100)

; förutsättningar och frågeställning skrivs ut då programmet körs!

; Vad vill du att programmet skriver ut för text inledningsvis?

;DelA

(text "Varje rektangel har omkretsen 200 cm" 18 "black")

(text "Värdet i rektanglarna är rektangelns area" 18 "black")

; Konstruktion av funktions som ritar rektangelbild med verktyget rectangle:

; På rad 14-15 ska du definiera funktionen med namnet number->square som ska tillverka(rita)en grön rektangel där en sida är s och den andra sidan sådan att omkretsen totalt blir 200 cm, hur lång är den andra sidan då?

;del B

(define (number->square s)

(rectangle s (- 100 s) "solid" "green"))

;Konstruktion av funktionen rektangelarea som beräknar arean av rektangeln som ritas ut

;På följande rad ska du definiera funktionen rektangelarea som beräknar arean där en sida är s

;del C

(define (rektangelarea s) (* s (- 100 s)))

; funktionen space på rad 22 används enbart för att ge mellanrum mellan de olika rektanglarna som ritas ut vid sidan om varandra.

;del D

(define (space d) (rectangle d 100 "solid" "white"))

; funktionen rektmedarea används för att montera samman rektangelbild med areavärde så att rektangelbilden läggs under rektangelareavärdet genom att använda vertyget underlay.

Areavärdet skrivs ut som en sträng och måste konverteras från numeriskt värde till sträng via number->string samt verktyget för text som är just text och som behöver 3 argument: sträng, storlek samt färg

```
(define (rektmedarea s) (underlay (frame (number->square s)) (text (number->string (rektangelarea s)) 12 "black" ) ))
```

; Här(rad 31) ritas samtliga rektanglar ur tabellen ut vid sidan om varandra. För att göra detta används bildverktyget beside. Du ska mata in de s-värde som finns i tabellen på reskektive ställe för funktionen rektmedarea totalt 14 st.

```
(beside (rektmedarea 5) (rektmedarea 7) (space 2) (rektmedarea 10) (space 2) (rektmedarea 15) (space 2) (rektmedarea 20) (space 2) (rektmedarea 30) (space 2) (rektmedarea 40) (space 2) (rektmedarea 50) (space 2) (rektmedarea 60) (space 2) (rektmedarea 70) (space 2) (rektmedarea 80) (space 2) (rektmedarea 90) (rektmedarea 93) (rektmedarea 95) )
```

; För att följa hur arean ändras mer stegvis då sidan s ändras använder vi en big-bangfunktion. Detaljerna hur en big-bangfunktion är uppbyggd ligger utanför denna kurs. Det som ritas ut är rektmedarea som ändras genom knapparna up(pil-upp) och down(pil-ner) steg för steg då man trycker ner dessa. Man måste börja med ´pil-upp annars havererar big-bang.

```
(define (stop? world)
  (> world 99))
(big-bang 0
        [to-draw rektmedarea]
        [stop-when stop?]
        [on-key handle-key ] )
(define (handle-key x k)
  (cond [ (key=? k "up") (add1 x)]
        [ (key=? k "down") (sub1 x) ]
        [ else x]))
```

Utvecklingstips:

Tröjförsäljning programmera.

Plotfunktion i scheme.

```
(place-image/align (triangle 48 "solid" "yellowgreen")
                   64 64 "right" "bottom"
                   (rectangle 64 64 "solid" "mediumgoldenrod"))
```