1. Övningar prioriteringsregler

4 + 2) - 10/5

(4 - 5) + (5 * 8)

(4 + (9 - 8)) * 5

-15 / 5 + -8

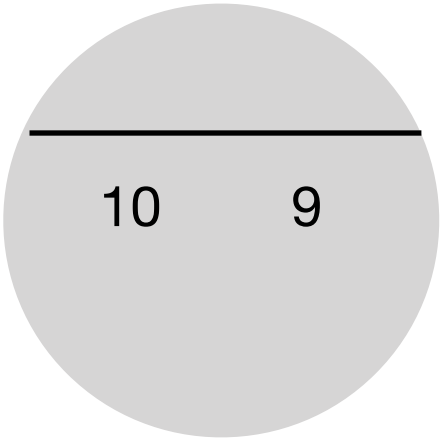
15 / 10 + (4 * -2)

(5 * 9) - (2 + 7)

2. Räknecirklar

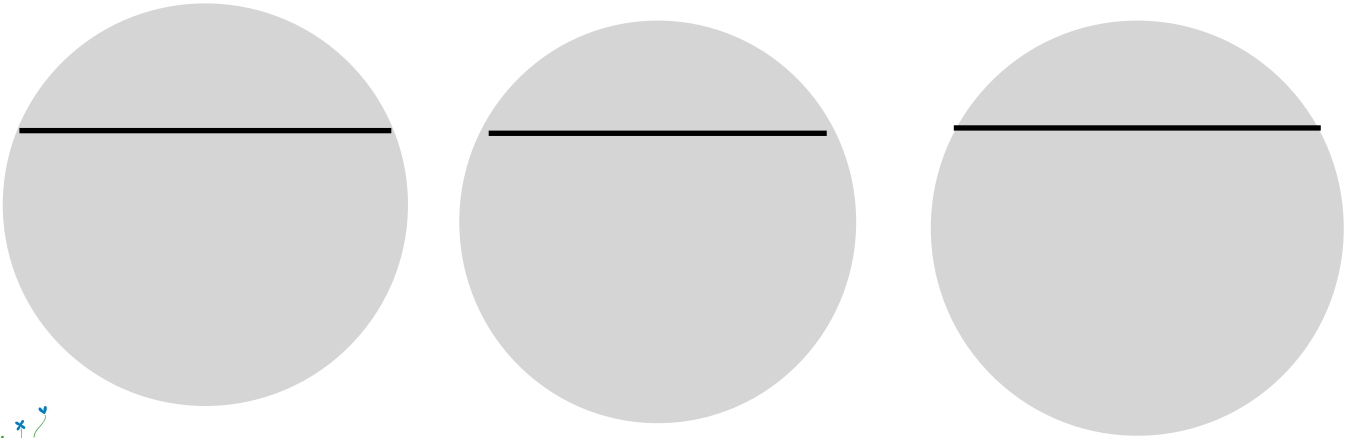
Ett sätt att ange ordningsföljden i ett uttryck är att först rita uttrycket som ett diagram. Det diagrammet vi ska använda oss av kallas räknecirkel. Dessa cirklar visar strukturen som händer inom ett uttryck, vilket är viktigt att veta om man ska skriva en funktion som utför beräkningar i WeScheme. Alla cirklar har två regler:

1. Varje cirkel måste ha en funktion som ligger högst upp i cirkeln.
2. Numren är skrivna nedan, i ordning från vänster till höger.

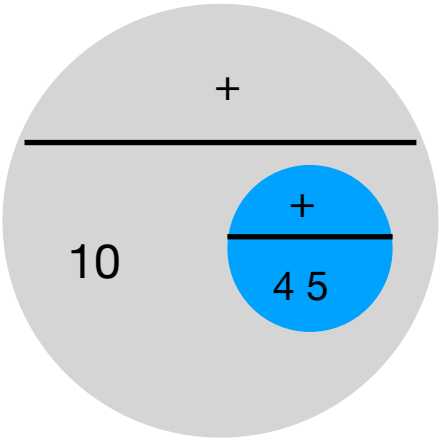


Ovan är ett exempel på hur det kan se ut. Cirkeln visar hur 10 + 9 skrivs.
Prova att göra egna utvärderingscirklar för följande uttryck:

1. 7 * 8
2. 10 - 18
3. 49 / 7



För att använda flera funktioner i samma uttryck kan vi kombinera utvärderingscirklar.



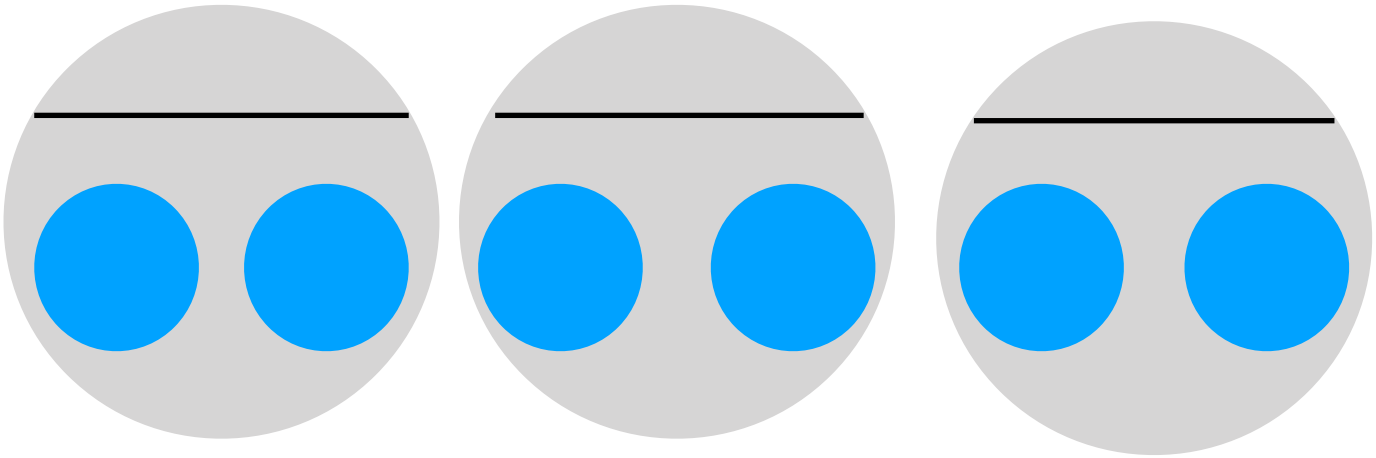
Hur skulle det här uttrycket se ut om du skrev det som ett matematiskt uttryck?

Skriv följande uttryck i utvärderingscirklarna.

9 * (17 + 2)

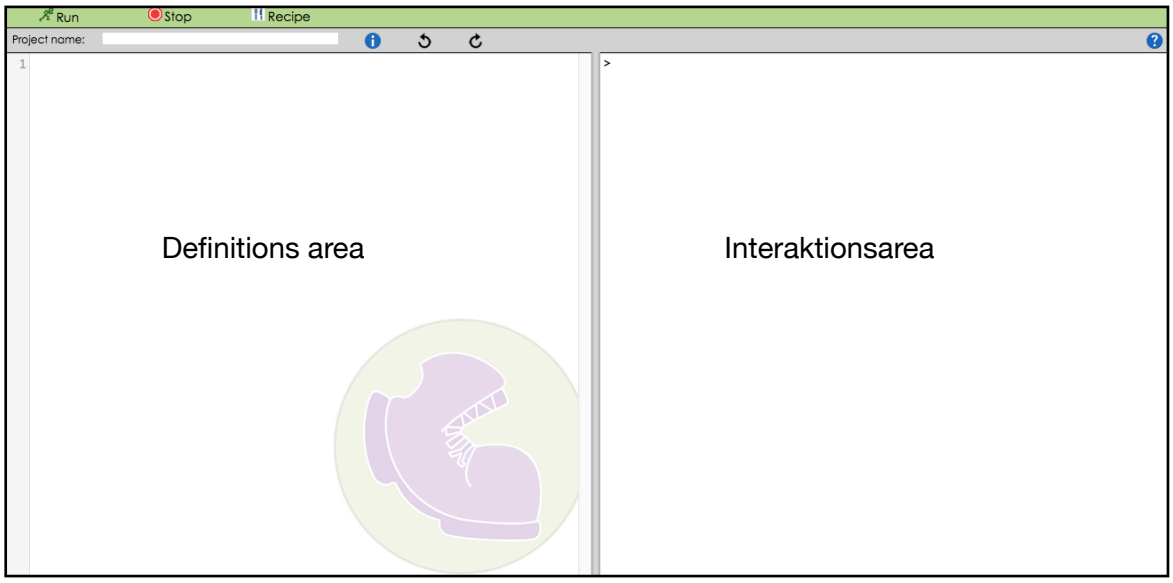
7 / 7 * (9 + 8)

(1 + 5) / (7 - 4)



3. Dags att börja programmera i Wescheme

Vi kommer att använda ett program som heter wescheme och är webbaserat. Du behöver inget konto för att kunna programmera.
När du kommer in så kommer det att se ut så här. Det som man möts av kallas editor och visas nedan. Här finns två stora rutor: definitions area och interactions area. Definitionsområdet är där programmerare definierar värden och funktioner i sitt program, medan området interaktioner gör att de kan experimentera med dessa värden och funktioner.



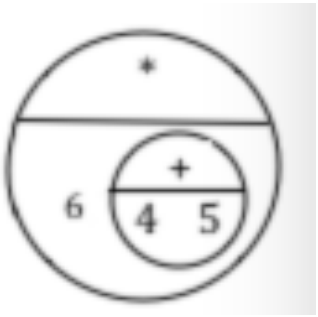
När ett program körs producerar det ett värde. Precis som när man skriver meningar så finns det vissa regler som bestämmer om det man skriver verkar vettigt. Program har också regler.

Regel nummer 1. Alla värden är riktiga uttryck.

Prova att skriva siffran 7 eller något annat tal i interaktionsarean för att se vad som händer. Vad händer om du skriver ditt namn?

Felmeddelanden som kommer upp är användbara för programmerare. Snarare än att säga "det här programmet fungerar inte", gör datorn det bästa för att berätta vad som gick fel och att ge dig så mycket information som möjligt för att hjälpa dig att åtgärda problemet. Så se till att du alltid läser dessa meddelanden noggrant! För de kan verkligen hjälpa dig att gå vidare.

Regel nummer 2. Varje öppen parentes följs av en funktion, därefter med ett eller flera riktiga uttryck, och slutligen av en slutlig parentes.



(* 6 (+ 4 5))

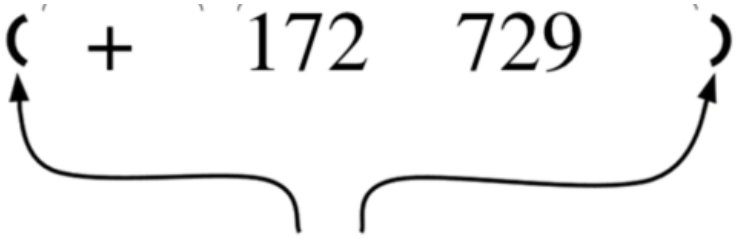
Dina utvärderingscirklar är lätta att omsätta i WeScheme.

För att översätta en räknecirkel till ett program, börja med en öppen parentes (och sedan funktionen som skrivs högst upp i cirkeln. Översätt sedan inmatningarna från vänster till höger på samma sätt och lägg till en avslutande parentes) Prova gärna att översätta de räknecirklar du gjorde innan till kod och skriv in dem i interaktionsarean.

Alla uttryck som kommer efter namnet på funktionen kallas argument.

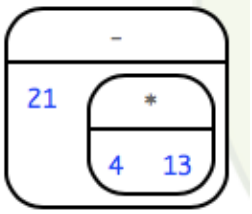
Namn, kommer alltid direkt efter den första öppna parentesen.

Argument, vad funktionen ska göra.

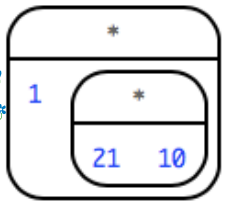


Varje funktion har en parantes som öppnar och stänger uttrycket.

Skriv följande i kod. Först på pappret sedan i interaktionsarean i WeScheme



_____



_____

4. Skriva kontrakt för program

Tänk efter vilken typ av data som programmet förväntar sig. Det kan vara flera typer av data och flera av samma sort.

Tex så ser kontraktet för en triangel ut så här:

rektangelgul: *nummer nummer string string* -> *image*
name domain range

Dvs programmet förväntar sig en längd, en bredd, om den ska vara fylld av färg eller ej samt färg på rektangeln. Programmet ser ut så här:

(rectangle (10 5 ”solid” ”blue”))

Skriv kontrakt för följande program:

divide producerar kvoten av två tal

_____ : _____ -> _____
name domain range

circle producerar en röd cirkel med radien 15

_____ : _____ -> _____
name domain range

1string producerar en ny sträng (= text) där bara den första bokstaven i en sträng (=ordet) finns med.

_____ : _____ -> _____
name domain range

Programmet add producerar summan av två tal.

_____ : _____ -> _____
name domain range

5. Programmera bilar

En bil
Kör programmet som heter ”Elev bil nytt” som ligger i WeScheme via följande länk. Välj edit. Då ser du programmet. Välj run i menyn för att starta programmet.

Elev bil

Bilen kör från ena sidan till den andra.

Ändra hastigheten i programmet och se vad som händer. Hastigheten finns i funktionen som heter car1-speed. (Innan du lämnar programmet, återställ hastigheten till 3)

Vad händer?

_____

_____

Stannar bilen på samma position oavsett hastighet?

_____

I programmet rör sig bilen i x-led med en hastighet av tre pixlar per klocktick, Det går 28 klocktick på en sekund.

Hur många pixlar rör sig bilen i x-led per sekund?

_____

Bilen börjar att åka vid tid tid 0. I programmet finns en ”BACKGROUND”. Den är en rektangel med en bredd och längd (rectangle (50 600 ”solid” ”green”)). Hur lång tid tar det för bilen att färdas från kant till kant? Räkna först ut och testa sedan genom att köra programmet och mäta tiden med tex en telefon.

_____

Titta på funktionen ”crash t ”. Den definierar när bilen ska stanna. Försök att förklara med ord vad funktionen gör och skriv det med text ovanför funktionen.
Börja skriv på samma rad som den rosamarkerade texten.

Förändra i villkoret så att du kan du få bilen att stanna innan trädet.

Två bilar
Invänta genomgång av lärare!

Öppna programmet ”Två bilar” genom att använda länken nedan och välj ”Edit” så att du kan se programmet.

Två bilar

Definiera bilarnas position i programmet genom att ange var bilen befinner sig i x-led, car1-pos samt car2-pos, vid tiden t. Som ledtråd har du _ där det ska stå något. Parenteserna är redan utsatta. Bilarna börjar sin färd vid med var sin hastighet och var sitt x-värde: CAR1-X0 samt CAR2-X0. Tänk på att använda dig av formeln s = v * t !

```
(define (car1-pos t) ( + CAR1-X0 ( * _ _) )
  )
(define (car2-pos t) ( + CAR2-X0 ( * _ _ ))
  )
```

Förändra hastigheten på CAR1-SPEED och CAR2-SPEED i programmet genom att byta ut siffrorna. Testa med både negativa och positiva tal. När du har testat klart bestämmer du vilken hastighet dina bilar ska ha.De ska gå "lagom" fort, dvs du ska kunna följa deras färd, samt ha olika hastigheter och börja från var sin kant.

```
(define CAR1-SPEED 3)
(define CAR2-SPEED -5 )
```

Kollision

Det är ju inte så bra att de kolliderar.

När kolliderar bilarna? Ange svaret i sekunder.

_____

Förändra värdet på stopp-villkoret (crash t) så att bilarna inte kolliderar! Använd dig av de metoder och lösningsstrategier som du har lärt dig.

Testkör ditt program och se om du kan undvika en bilkrasch!