

### Variabler och strängar

```
(println "Hej världen!")           ;; Skriv ut "Hej världen!"  
  
(def meddelande "Hej världen!")    ;; Definiera variabel  
(println meddelande)              ;; Skriv ut värdet
```

### Matematiska operatörer

|                |             |                    |
|----------------|-------------|--------------------|
| Addition       | (+ 5 3)     | 8                  |
| Subtraktion    | (- 5 3)     | 2                  |
| Multiplikation | (* 5 3)     | 15                 |
| Division       | (/ 5 3)     | 5/3                |
|                | (/ 5.0 3.0) | 1.6666666666666667 |
| Modulo         | (mod 5 3)   | 2                  |

### Jämförelse

|                          |             |
|--------------------------|-------------|
| lika med                 | (= x 42)    |
| inte lika med            | (not= x 42) |
| större än                | (> x 42)    |
| större än eller lika med | (>= x 42)   |
| mindre än                | (< x 42)    |
| mindre än eller lika med | (<= x 42)   |

### If-satser

```
(if (>= ålder 18)                  ;; Om villkoret stämmer  
  ("Du får rösta")                ;; kör första funktionen.  
  ("Du får inte rösta"))          ;; Annars kör andra.
```

### Listor

```
(def vänner                        ;; Definiera lista med strängar  
  ["Jonas" "Simon" "Johan" ])  
  
(first vänner)                    ;; Hämta första element i listan  
(last vänner)                     ;; Hämta sista element i listan  
(nth vänner n)                    ;; Hämta n:te elementet  
(rest vänner)                     ;; Alla element utom första  
(pop vänner)                      ;; Alla element utom sista  
(empty? vänner)                   ;; Kollar ifall listan är tom  
(count vänner)                     ;; Listans längd
```

Obs! Strängar kan ses som lista av  
karaktärer:

```
(empty? "Jonas")                  ;; Är strängen tom?  
(count "Jonas")                   ;; Hur lång är strängen?  
  
(assoc vänner 0 "Calle")          ;; Returnerar ["Calle" "Simon"  
  "Johan"]
```

### Listor (cont)

```
(def matris [["a"] ["b"] ["c"]])  ;; 2-dimensionell lista  
(assoc-in matris [1 0] "x")       ;; Returnerar [["a"] ["x"] ["c"]]
```

### Funktioner

```
(defn hälsa-utan-namn []           ;; Definiera funktion utan parameter  
  "Hej hej!")  
  
(hälsa-utan-namn)                 ;; Anropa funktion utan parameter  
  
(defn hälsa-med-namn [namn]      ;; Definiera funktion med parameter  
  (str "Hej" namn "!"))  
  
(hälsa-med-namn "Filip")         ;; Anropa funktion med parameter
```

### Rekursion

```
(defn foo []                       ;; Oändlig rekursion  
  (foo))  
  
(defn foo [tal]                  ;; Ha med basfall i rekursiva funktioner  
  (if (> tal 0)                   ;; Basfallet avgör om:  
      (foo (- tal 1))             ;; rekursionen fortsätter  
      (println "stop")))         ;; rekursionen avbryts
```

### Felsökningstips

1. Stava rätt  
variabelnamn kan lätt bli  
varibaelnamn
2. Kolla parenteser  
Se till att varje ( matchas med en )  
Eller [ med ]
3. Stängda strängar?  
"Detta kommer inte funka."  
Ej heller det här."
4. Kontrollera att funktionen  
anropas!  
(foo)
5. Kontrollera antal parametrar  
(foo [a b] ;<- Tar 2 parametrar  
(println a )  
(println b ))  
  
(foo "en parameter" ) ;<- Fel antal