Rekursion Arbeitsweise

Demonstration der Arbeitsweise
einer **rekursiven** Funktion

am Beispiel
Anhängen eines Elements
an eine Liste

Rekursion Arbeitsweise

- Die Funktion:

```
(define
  (anhaengen element liste)
  (cond
    ((null? liste)
     (list element))
    (else
     (cons
      (first liste)
      (anhaengen element (rest liste))))
    )))
```

Rekursion Arbeitsweise

Funktionsaufruf:

(anhaengen 'f '(a b c d e))

element	liste				
f	a	b	c	d	e

Rekursion Arbeitsweise

Funktionsaufruf:

(anhaengen 'f '(b c d e))

element	liste			
f	b	c	d	e

Rekursion Arbeitsweise

Funktionsaufruf:

(anhaengen 'f '(c d e))

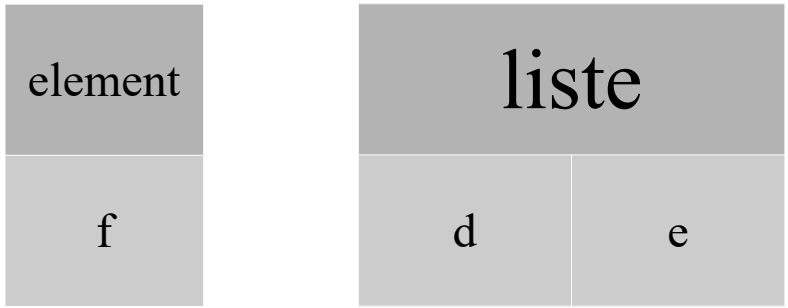
element
f

liste		
c	d	e

Rekursion Arbeitsweise

Funktionsaufruf:

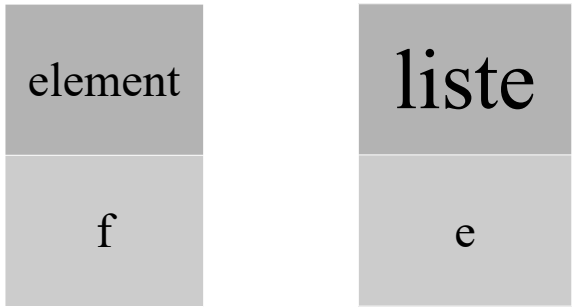
(anhaengen 'f '(d e))



Rekursion Arbeitsweise

Funktionsaufruf:

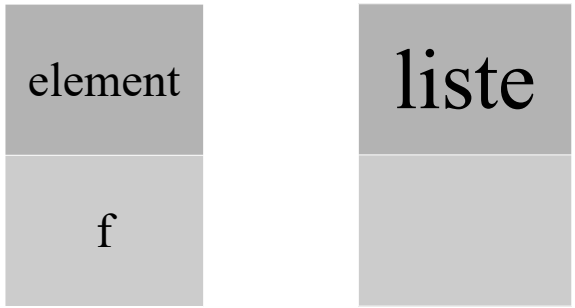
(anhaengen 'f '(e))



Rekursion Arbeitsweise

Funktionsaufruf:

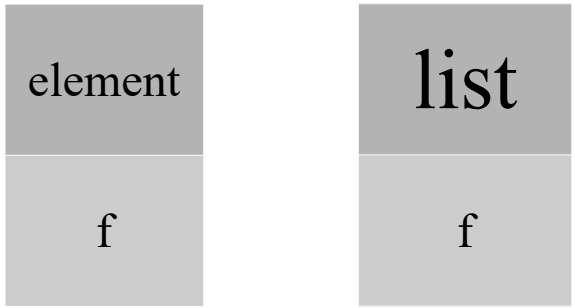
(anhaengen 'f '())



Rekursion Arbeitsweise

Auswertung:

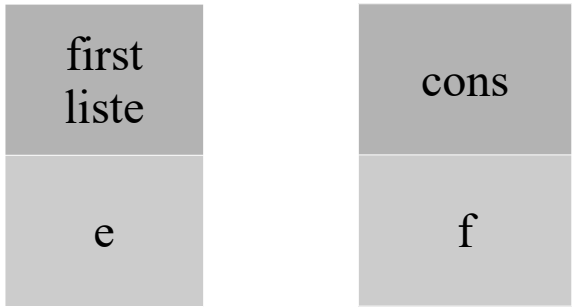
(list 'f)



Rekursion Arbeitsweise

Auswertung:

(cons 'e (anhaengen 'f '( ))))



Rekursion Arbeitsweise

Auswertung:

(cons 'd (anhaengen 'f '(e)))

Rekursion Arbeitsweise

Auswertung:

(cons 'c (anhaengen 'f '(d e)))

first liste
c

cons		
d	e	f

Rekursion Arbeitsweise

Auswertung:

(cons 'b (anhaengen 'f '(c d e)))

first liste	cons			
b	c	d	e	f

Rekursion Arbeitsweise

Auswertung:

(cons 'a (anhaengen 'f '(b c d e)))

first liste	cons				
a	b	c	d	e	f

Rekursion Arbeitsweise

Funktionsergebnis ausgeben

(anhaengen 'f ' (a b c d e))

liste					
a	b	c	d	e	f