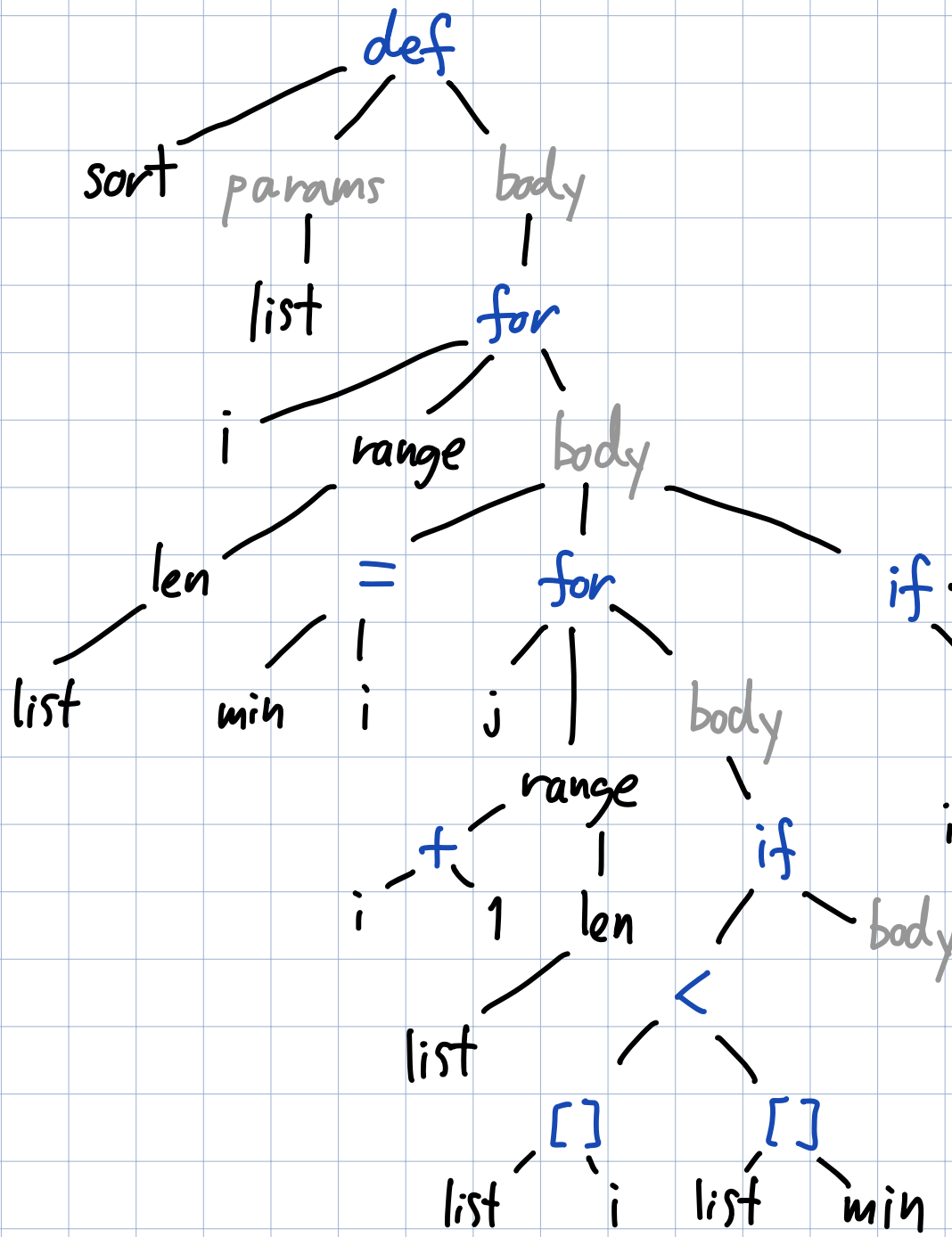
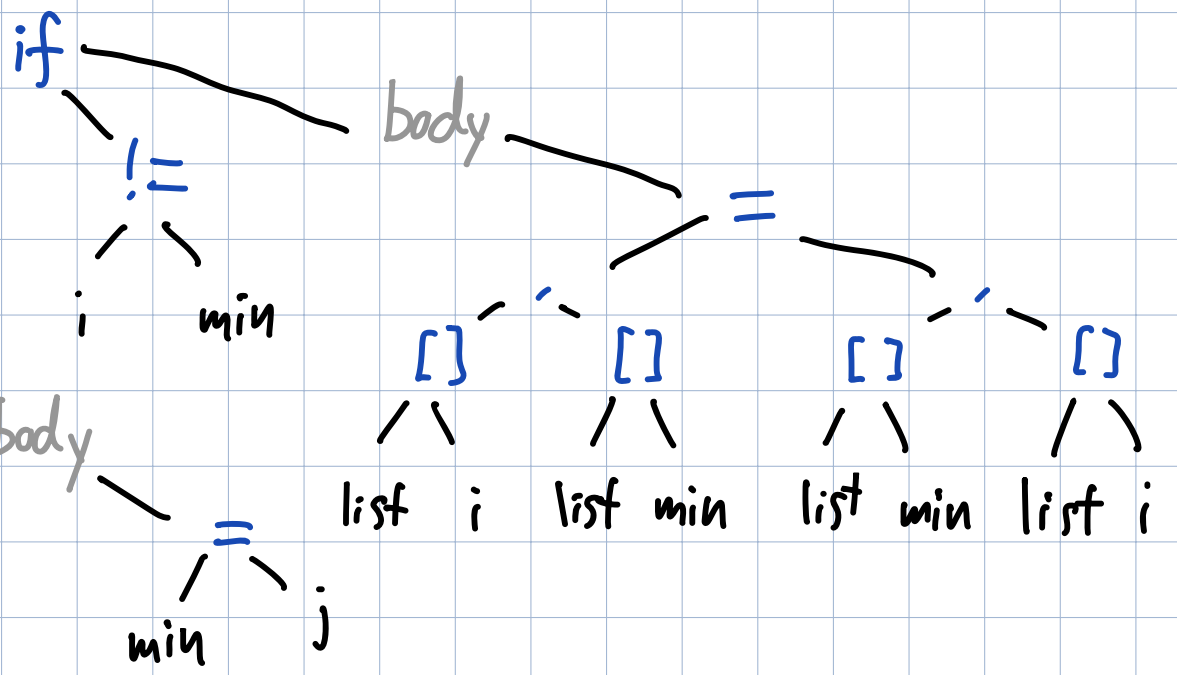
Programmeringsspråk er trøtt

```
def sort(list):
    for i in range(len(list)):
        min = i
        for j in range(i + 1, len(list)):
            if list[j] < list[min]:
                min = j
        if i != min:
            list[i], list[min] = list[min], list[i]
```

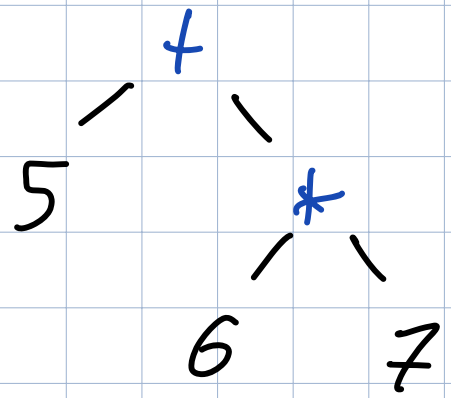
```
def sort(list):
    for i in range(len(list)):
        min = i
        for j in range(i + 1, len(list)):
            if list[j] < list[min]:
                min = j
        if i != min:
            list[i], list[min] = list[min], list[i]
```





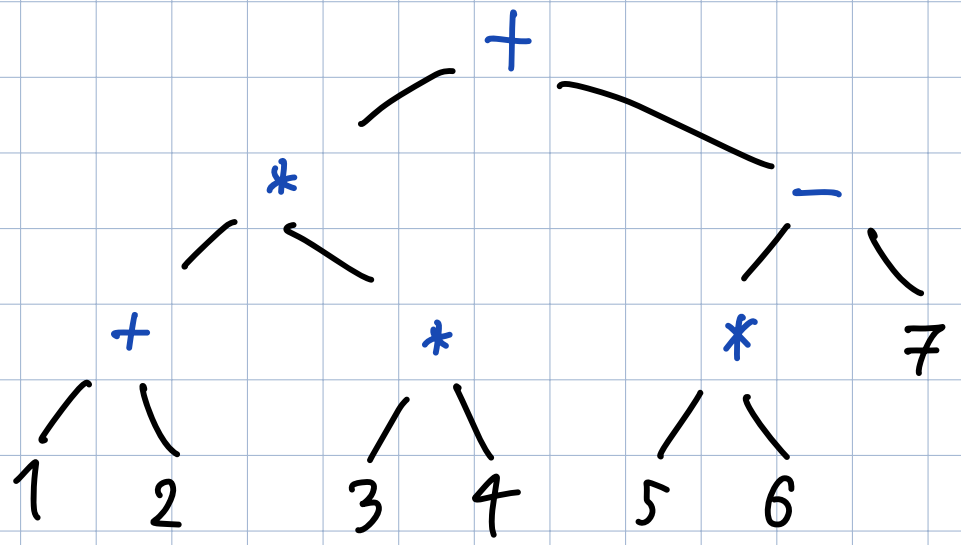
$5 + (6 * 7)$

```
{ op: +,
  left: 5,
  right: { op: *,
           left: 6,
           right: 7}
}
```



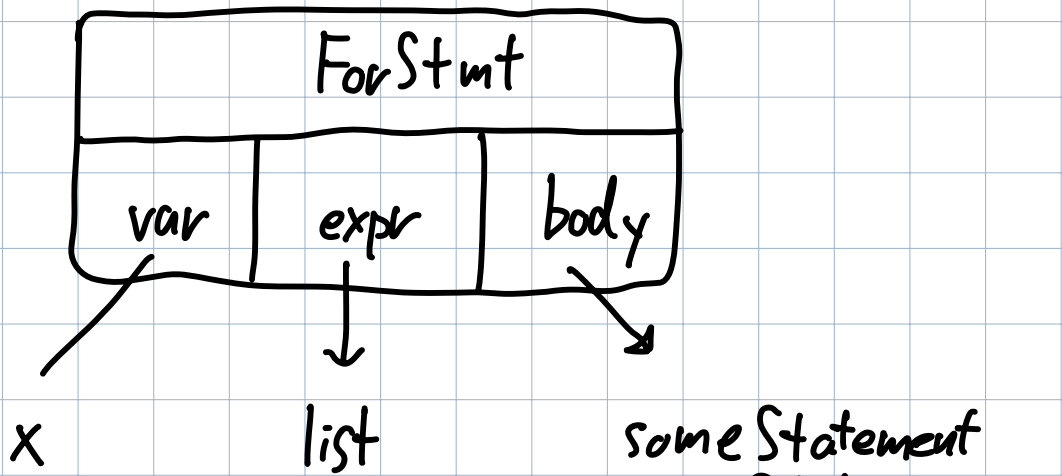
```
def eval_expr(expr):
    if is_number(expr):
        return expr
    l = eval_expr(expr.left)
    r = eval_expr(expr.right)
    return expr.op(l, r)
```

$$(((1+2)*(3*4))+((5*6)-7))$$



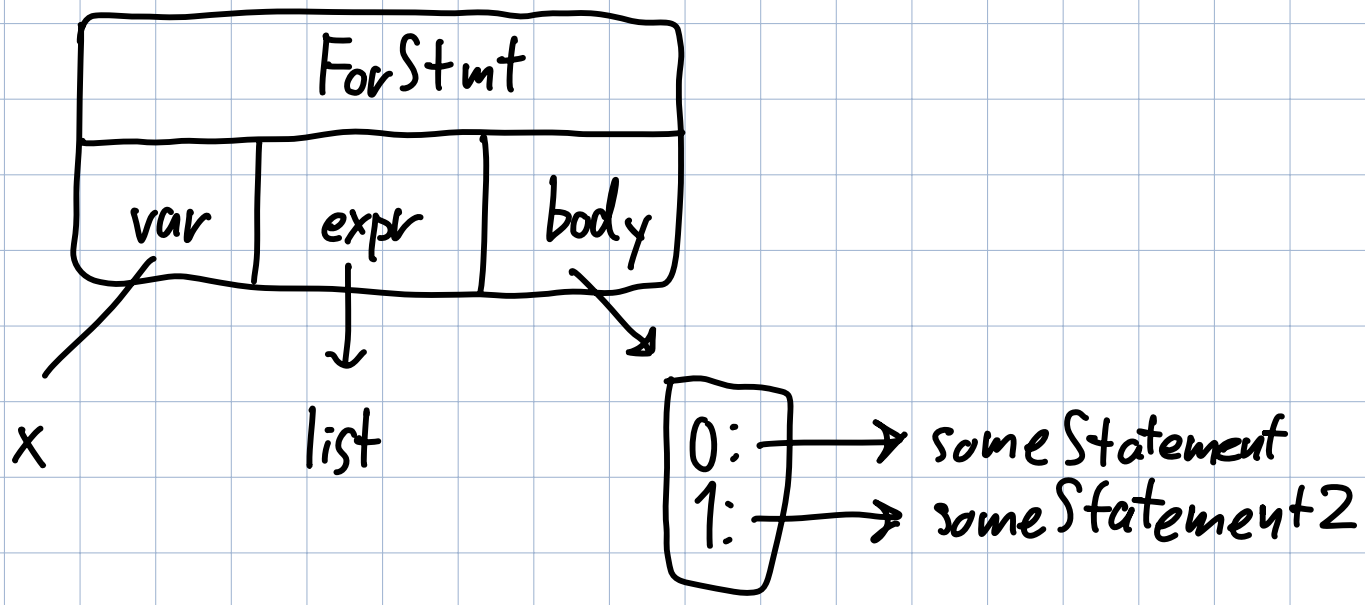
```
for x in list:
    someStatement
    someStatement2
```

```
class ForStmt:
    var : Variable
    expr : Expression
    body: Statements
```

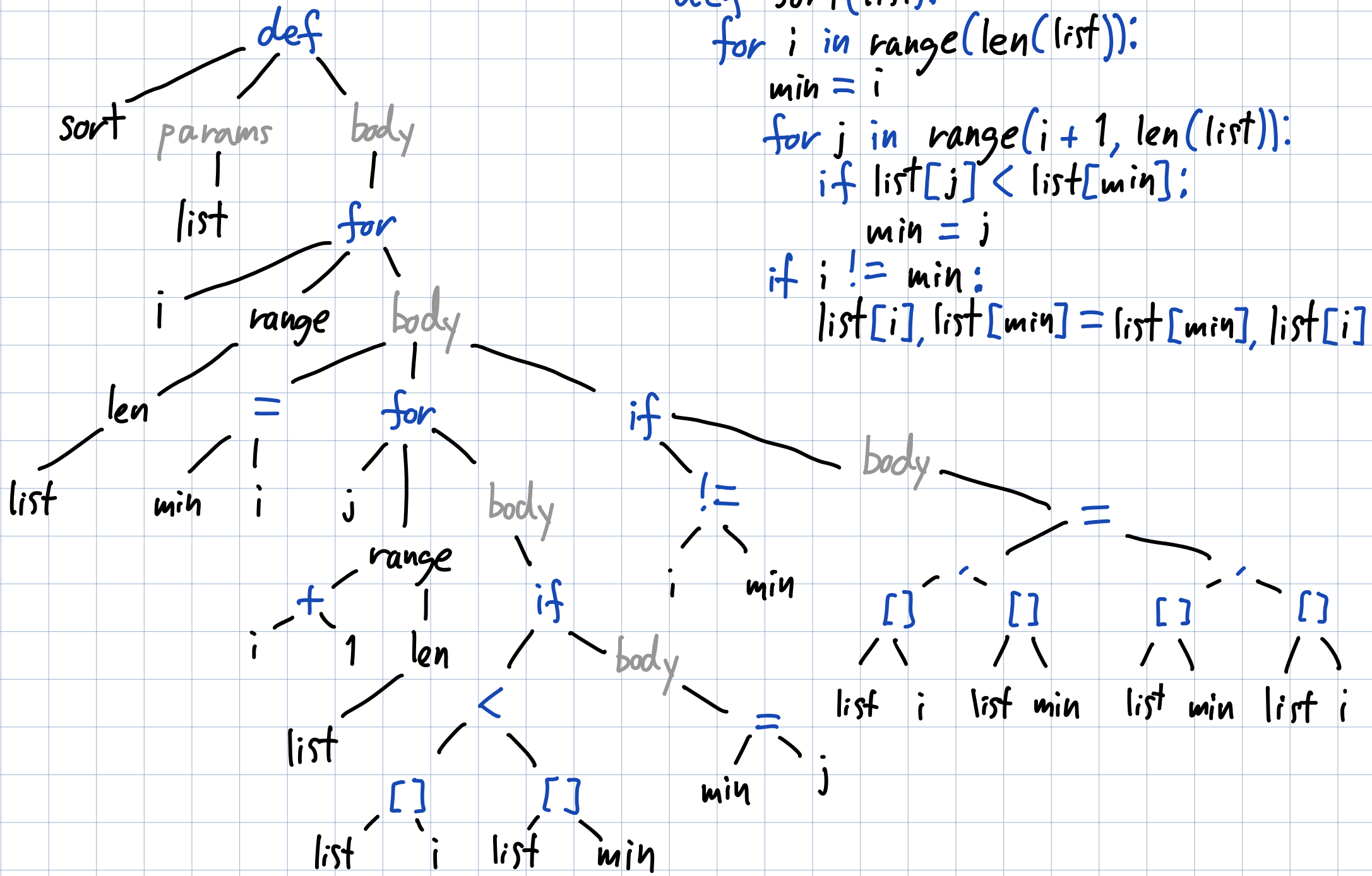


```
for x in list:
    someStatement
    someStatement2
```

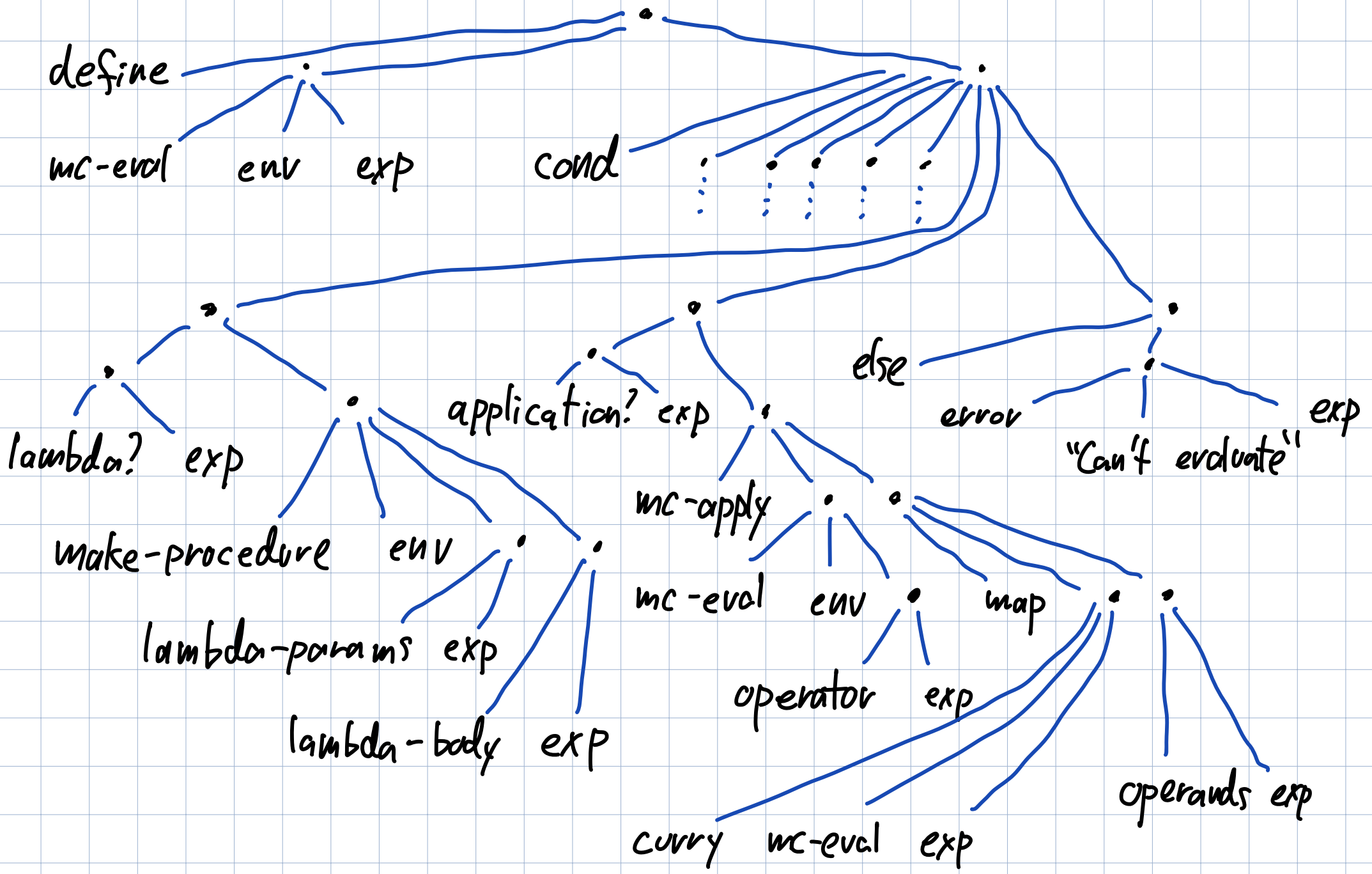
```
class ForStmt:
    var : Variable
    expr : Expression
    body: Statements
```



```
def sort(list):
    for i in range(len(list)):
        min = i
        for j in range(i + 1, len(list)):
            if list[j] < list[min]:
                min = j
        if i != min:
            list[i], list[min] = list[min], list[i]
```



```
(define (mc-eval env exp)
  (cond ((self-evaluating? exp) exp)
        ((variable? exp) (lookup-variable-value env exp))
        ((definition? exp) (eval-definition env exp))
        ((if? exp) (eval-if env exp))
        ((begin? exp) (eval-begin env exp))
        ((lambda? exp) (make-procedure
                        env
                        (lambda-params exp)
                        (lambda-body exp)))
        ((application? exp) (mc-apply
                             (mc-eval env (operator exp))
                             (map (curry mc-eval env) (operands exp))))
        (else (error "Can't evaluate" exp))))
```


define
mc-eval
env
exp
cond
lambda?
exp
make-procedure
env
lambda-params
exp
lambda-body
exp
application?
exp
mc-apply
mc-eval
env
operator
exp
map
curry
mc-eval
exp
operands
exp
else
error
"Can't evaluate"
exp