Scheme-like Languages

concept	like	unlike
very simple syntax: programs straightforwardly as data		C++
objects are allocated dynamically, never forward (GC)	Java, ML	C++
dynamic type checking	Python	Java, C++, ML
static scoping	Python, ML, C++, Java	Lisp
call by value	ML, Java, Python C++(mostly)	C++ &

- objects include the usual (nums, strings, etc.)
- high-level arithmetic
- TRO - tail recursion optimization

```
C++
int factorial (int n) {
    return (n == 0 ? 1 : n * factorial (n - 1));
}

Lisp
(define (factorial n)
    (if (zero ? n) 1
        (*factorial (- n 1) n))
)

^ bad way to write, grows the stack too much
(define (fact n a)
    if (zero ? n) a
    (fact (- n 1) (* a n)))

(define (factorial n)
    fact (n 1))
```

tail - calls a function then immediately returns what the function returns

- the last tail call does not grow the stack

downside of dynamic scoping:

- more runtime costs
- software reliability

named `let` - gives tail recursive call

Scheme does static scoping before it expands the macro