Many points of view

• Before we dive into object-oriented programming, let’s look at a couple of other points of view.

• We will take a quick look at a couple of other ways of programming and few different development environments.

• Don’t worry about getting every detail at this stage, this is more just to get an impression.
This is a C program to compute factorials, 
\[ n! = n \times (n - 1) \times \ldots \times 3 \times 2 \times 1. \]

```c
int factorial(int n) {
    int i, prod;
    prod = 1;
    for (i = 1; i <= n; i++)
        prod = prod * i;
    return prod;
}
```

The main things to notice are the assignments and the loop.
This is a Scheme program to compute n!

```
(define factorial (lambda (n)
    (if (= n 1)
        1
        (* n (factorial (- n 1)))
    )))
```

Scheme is a member of the Lisp family of languages, the first of which from about 1960.

It *uniformly* uses the syntax `(operator arg1 arg2 ...)`.

This program uses *recursion* and the fact that, for $n > 1$, $n! = n \times (n - 1)!$
Uniformity vs Convenience

• Which is better, to have convenient, but irregular syntax, like \(1 + \cos(\theta)\),

or to have a completely uniform, but less convenient syntax, like \((+ 1 (\cos \theta))\).

• The irregular syntax is easier to use and the regular syntax is easier to compose.

• For single jobs a Swiss Army knife is better than a Lego brick, but \textit{how would you make a replica of the Empire State building out of Swiss Army knives.}
The Lisp family of languages

- Most things are made up of “cons” cells that contain two things.
- The things they contain can be values or pointers to other things.
- Can make complicated data structures from these.
The Eclipse IDE for Java Development

```java
class Hello {
    public static void main(String[] args) {
        // Create array and initialize all entries.
        boolean[] marks = new boolean[80];
        for (int i = 0; i < marks.length; i++)
            marks[i] = true;
        // For each number k >= 2
        for (int k = 2; k < marks.length; k++) {
            // If it is prime, cross out its multiples.
            if (marks[k] == true)
                for (int j = 2; j * k < marks.length; j++)
                    marks[i * k] = false;
        }
        System.out.print("The primes up to 79 are: ");
        System.out.print(" + ");
        for (int i = 2; i < marks.length; i++)
            if (marks[i] == true)
                System.out.print(" + i");
        System.out.println(" ");
    }
}
```
The DrRacket IDE for Scheme

```
(define n 6)

(define factorial (lambda (n)
    (if (= n 1) 1 (* n (factorial (- n 1))))))
```

Welcome to DrRacket, version 5.1.1 [3m].
Language: R5RS [custom]; memory limit: 128 MB.
> (+ n n)
12
> (factorial n)
720
>
The *Maple* Environment

```maple
> f := sin(x^2) + cos(x);

> diff(f, x);

> fsolve(f, x);

> fact := proc(n) if n = 1 then 1 else n*fact(n-1) fi end:
> fact(50);
> fact(300);
```

```
f := sin(x^2) + cos(x)

2 cos(x^2) x - sin(x)

4.686736151

3041409320171337804361260816606476884377641568960512000000000000

306057512216440636035370461297268629388588041735769994167767412594765331767168674

...