Scheme

- Objects in scheme (every value in scheme is a value)
    - In this sense it is object oriented
    - In another sense it isn't, because it doesnt have classes etc.
    - Objects are allocated dynamically and are never freed
        - Like Java, Python, OCaml
        - Unlike C
    - Types are latent (dynamic type checking), not manifest (static type checking)
        - +Python, -Java, Ocaml, C, C++
    - On the other end of the spectrum -> uses static scoping
        - Can be determined at compile time
            - Like Java, Python, C, OCaml
        - Examples:
        - Lisp Dynamic scoping -> look in current fn, then in caller, then in caller's caller's
            - Other systems also do it, PATH, sh environment
            - You can set a variable in your program and can affect a lot of other programs -> get extra flexibility
        - Static scoping, look in current fn, then in definer, then in definer's definer
            - More efficient, predictable, reliable
        - Static chain is looking at definer chain, while dynamic is looking in the caller
- Wide variety of built in object types, which include procedures
- Very simple syntax, with a straightforward representation as data

QUESTION: WHAT ARE SIDE EFFECTS? (sorry austin im dumb so i dont even know what those mean)

- Tail recursion optimization is required, to avoid blowing the stack when you have deeply involved recursion
    - If the last thing your function does is call another function, and returns whatever g returns, then the compiler is REQUIRED to optimize the call in such a way that we do not grow the stack