Lecture-02-CMSC330:

OCaml:

- In functional programming, everything is immutable.
- Functions do not have side effects.
- **Type inference:** the compiler can determine the return type of a function/expression based on the input type.
- Harder to step through and debug code in OCaml, more reliance on print statements.

Lists in OCaml:

- All items in a list must have the same type. Lists are immutable.
- Can insert lists into lists.
- Methods of creating a list:
    1. [ ];; - empty list, [ ]
    2. 1::[ ];; - [1]
    3. 1::2::3::[ ];; - [1, 2, 3]
    4. [1;2;3;4];; - [1, 2, 3, 4]
- To append an int/int list reference, the next reference must be of type list.

ex: ``` let app x y = 1:: x :: y;; ``` = int -> int list -> int list

ex: ``` let app x y = x :: y;; ``` = 'a -> 'a list -> 'a list

ex: ``` let app (x: int list) y = x :: y;; ``` = int list -> int list list -> int list list