Lecture-01-CMSC330:

- A language is **Turing complete** if it can compute any function computable by a Turing machine. Nearly all programming languages are Turing complete.

Language Attributes:

- Syntax: What a program looks like
- Semantics: What a program means (mathematically), i.e. what it computes
- Paradigm and Pragmatics: How programs tend to be expressed in the language
- Implementation: How a program executes (on a real machine)

OCaml:

- A mostly functional language. Natural support for **pattern matching**. Has full featured **module system**. Includes **type inference**.

Zero-cost Abstractions in C/C++ and Rust:

- A key motivator for writing code in C and C++ is the low/zero cost of the abstractions used. Data is represented minimally, no metadata required. Stack-allocated memory can be freed quickly. No garbage collector or other mechanisms are needed.
- However, no-cost abstractions in C/C++ are insecure.
- Rust has safe, zero-cost abstractions through the type system's use of ownership and lifetimes.

Implementation:

- Can implement a programming language through **compilation** or **interpretation**.

Compilation:

- Source program is translated ("compiled") to another language, traditionally directly executable machine code.

Interpretation:

- Interpreter executes each instruction in source program one step at a time, no separate executable.

OCaml Compiler:

- OCaml programs can be compiled using ocamlc, producing .cmo ("compiled object") and .cmi ("compiled interface") files. Can also compile with ocamlopt, to produce .cmx files, which contain native code and are faster but not platform-independent.
- Use -o to set output file name.
- Use -c to compile only to .cmo/.cmi and not to link.
- Can compile multiple files together ("ocamlc util.ml main.ml") or compile separately ("ocamlc -c util.ml", "ocamlc util.cmo main.ml").

OCaml Top-Level:

- The **top-level** is a read-eval-print loop (REPL) for OCaml, started via the **ocaml** command. **utop** is an alternative top-level, which improves on **ocaml**.
- Exit **top-level** with Control-D or by calling exit 0.

OPAM: OCaml Package Manager:

- **opam** is the package manager for OCaml.
- Install the following packages with **opam:** ounit (a testing framework similar to minitest), utop (a top-level interface), dune (a build system for larger projects).

Project Builds with **dune**:

- **dune** is used to compile projects, automatically finding dependencies, invoking the compiler and linker. Must define a **dune** file, similar to a Makefile in C.
- Run a project's test suite with **dune runtest**.
- Load modules defined in src/ into the **utop** top-level interface: **dune utop drc**

Notes on **;;**:

- ;; ends an expression in the top-level of OCaml, used to say "Give me the value of this expression".
- Not used in the body of a function.
- Not needed after each function definition.