- Announcement ① Cheatsheet available on webpage
- Announcement ② Look at weekend reading material on the website.

Plan for today

- Ocaml as calculator
- "Hello, World" : How to compile Ocaml files.
- Variable bindings / cf. mutability in Python
- Evaluation rules / scoping

Aspects of a programming language

- Syntax : What programs "look like"
- Semantics : What programs "mean".
  - Evaluation.
  - Mental model reasonably accurate
- Coding idioms
- Libraries
- Tool : compilers, interpreters, debuggers, IDEs, bug finders, static analyzers etc.

- Language is expression oriented, rather than statement oriented.

- Compilation model.

- ocaml: Source code interpreter / REPL
  - ↓
  - utop is strictly superior

- ocamlc: Bytecode interpreter

  Source code —ocamlc→ Bytecode a.out

  ocamlc: Fastest to compile, slowest to run.

- ocamlopt: Native code compiler

  Source code —ocamlopt→ Native instructions a.out

  ocamlopt: Slowest to compile, Fastest to run

((2+3)-5)

↑

Parentheses disambiguate precedence.

- Unit vs nullptr_t

↑ Type of the value ()

↑ Type of the null pointer literal NULL/C nullptr/C++

int * x = nullptr;

("Hello!", 5)

↑

Parentheses define pairs & tuples

("Hello", 5, 'c')

string * int * char

()

↑

Parentheses define the unit type

The unit type contains only value (). (instance)

The type bool of values true, false

The type color = Red | Green | Blue of 3 values.

Food for thought: Does it make sense to define a type with no values?

- What's the difference between a calculator & a computer? → Turing machine.

- RAM — "Random access memory" "Random access machine"
- Instructions

"Von Neumann architecture" ← Single bank of memory having both data & the program.

"Harvard architecture"

- Having rich data-types

Observation: let y=3 in let y=2 in (y)+3 evaluates to 5.

Hypothesis: This y evaluates to 2.

Question: What is the underlying theory of "scoping"?