

- 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, <sup>static analyzers</sup> bug finders etc.

- Language is expression oriented, rather than statement oriented.

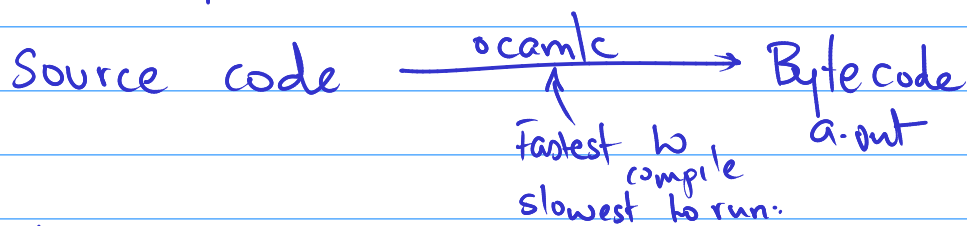
---

- Compilation model.

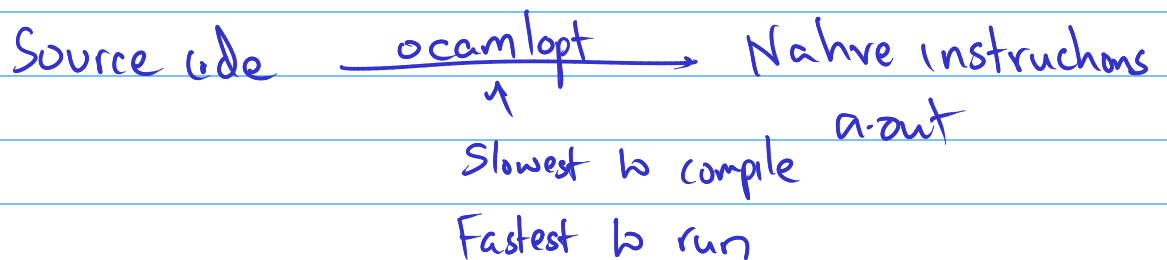
- ocaml: Source code interpreter / REPL

↓  
which is strictly superior

- ocamlc: Bytecode interpreter



- ocamlc.opt: Native code compiler



- Unit vs nullptr\_t

↑  
Type of  
the value ()

int \* x = nullptr;

("Hello", 5, 'c')

string \* int \* char

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

((2+3)-5)

↑  
Parentheses  
disambiguate  
precedence.

("Hello!", 5)

↑  
Parentheses define  
pairs & tuples

()

↑  
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

Single bank of memory ← "Von Neumann architecture"  
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"?