Exercise 0: Installing Ocaml on your machine (Ungraded exercise but recommended)

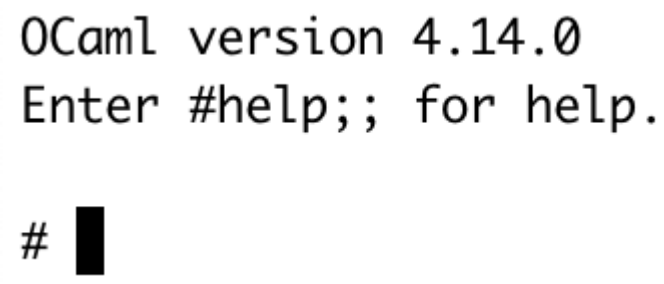
Try to install Ocaml on your machine. The most reliable, updated how-to is the official page:

https://ocaml.org/docs/up-and-running

After installation, you run

```
% ocaml
```

on your command line. Above, "%" refers to prompt of your terminal. So you only need to type "ocaml". You should get an ocaml toplevel. Below is the screenshot of the toplevel from my machine.



Exercise 1. Toplevel (2)

If you have finished Exercise 0, you can continue using Toplevel from your machine in this exercise.

If you have not finished Exercise 0, you can use an online Ocaml IDE from the site:

https://try.ocamlpro.com/

Find the "#" on your Toplevel. That is a prompt, which is not part of the Ocaml language. Try to type `42;;` in the Toplevel following "#". Your Toplevel may or may not generate double-semicolon `;;` for you.

Search on the Internet. Write in English about your understanding on the role of the double-semicolon `;;` in Ocaml. This is an open question.

Exercise 2. Expressions (54)

Build-in-types

Consider the following expressions:

1. 42
2. 42.1
3. "hello"
4. true
5. ()
6. 'a'

Type each expression at Toplevel followed by `;;` if it is not automatically generated. Check the evaluation result of the expression and the type, and fill in the table below.

Expression	type	evaluation

Functions

Consider the following expressions:

7. print_endline
8. fun x -> x + 1
9. fun x y -> x+y
10. (+)
11. (+.)
12. (^)

Type each expression at Toplevel followed by `;;` if it is not automatically generated. Check the evaluation result of the expression and the type, and fill in the table below. For functions, such as `fun x -> x * x`, you write the evaluation result as `<fun>`.

Expression	type	evaluation

Function Evaluations

Consider the following expressions:

13. (fun x -> x + 1) 3
14. (fun x y -> x+y) 2 3
15. (fun x y -> x+y) 2
16. let x = 3 in x + 1
17. let y = 3 in let x = 2 in x + y
18. print_endline "hi"

Type each expression at Toplevel followed by `;;` if it is not automatically generated. Check the evaluation result of the expression and the type, and fill in the table below.

Expression	type	evaluation

Exercise 3. Definitions (24)

Consider the following definitions.

1. let x = 3
2. let s = "hello"
3. let a = ()
4. let b = true
5. let f = fun x -> x + 1
6. let f x = x + 1
7. let f = fun x y -> x + y
8. let f x y = x + y

Type each definition at Toplevel, followed by `;;` if it is not automatically generated. Check which variable is defined, of what type, and evaluation result. Fill in the table below.

Definition	defined variables	type	evaluation

Definition	defined variables	type	evaluation

Exercise 4. Put it all together (20)

1. Define a function avg in Ocaml that computes the average of two float.
2. Complete the gcd function of type int -> int -> int which calculates the gcd using the Euclidean method, that is:

- gcd(u, 0) = u
- gcd(u, v) = gcd(v, u mod v) otherwise"

```
let rec gcd u v =
  0 (* To complete *)




(* some tests *)
let () = Printf.printf "%d\n" (gcd 8 12)
let () = Printf.printf "%d\n" (gcd 48 18)
```

Once you finish, copy the code to Toplevel and check out the result.

Feel free to search online. You may not understand everything in the code. That is fine; we will learn things incrementally.