

CSE216

Foundations of Computer Science

Instructor: Zhoulai Fu

State University of New York, Korea

Many slides taken from Cornell's CS3110. Thanks!
https://www.cs.cornell.edu/courses/cs3110/2014fa/lecture_notes.php

Plan

- A review exercise
- Option

Review exercises on lists

Good exercises from “99 Problems in OCaml”

<https://v2.ocaml.org/learn/tutorials/99problems.html>

0

- Find the number of elements of a list.

1

- Find the `n`_th element of a list. If `n` is smaller than list length, fail with error. Similar with `List.nth`

```
# List.nth ["a"; "b"; "c"; "d"; "e"] 2;;  
- : string = "c"  
# List.nth ["a"] 2;;  
Exception: Failure "nth".
```

3

- Eliminate consecutive duplicates of list elements.

```
# compress ["a"; "a"; "a"; "a"; "b"; "c"; "c"; "a"; "a"; "d"; "e"; "e"; "e"; "e"];;  
- : string list = ["a"; "b"; "c"; "a"; "d"; "e"]
```

4

- Duplicate the elements of a list.

```
# duplicate ["a"; "b"; "c"; "c"; "d"];;  
- : string list = ["a"; "a"; "b"; "b"; "c"; "c"; "c"; "c"; "d"; "d"]
```

5

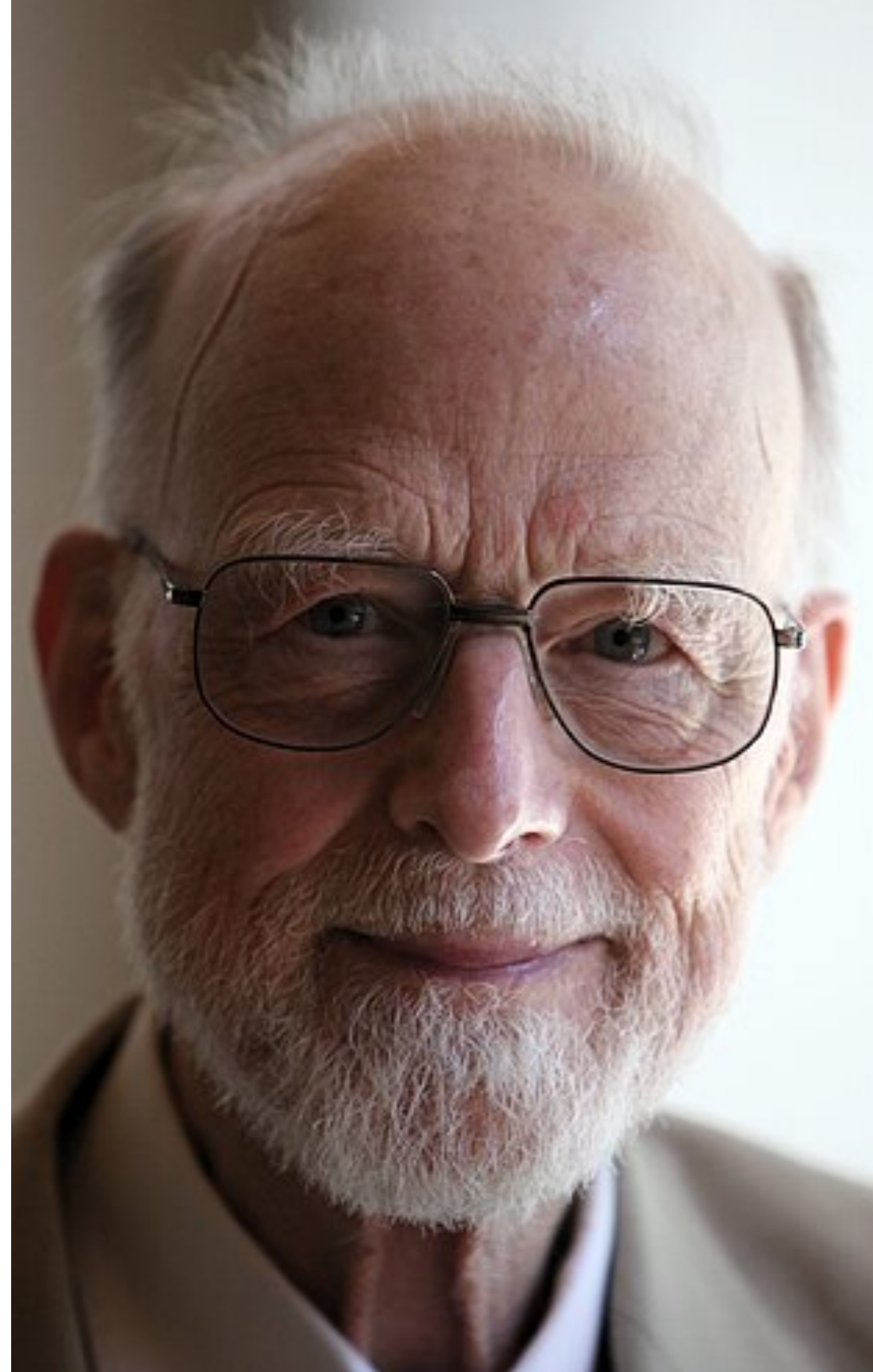
- Replicate the elements of a list a given number of times.

```
# replicate ["a"; "b"; "c"] 3;;  
- : string list = ["a"; "a"; "a"; "b"; "b"; "b"; "c"; "c"; "c"]
```


Option

“I call it my billion-dollar mistake...
the invention of the null reference.
It has led to countless errors,
vulnerabilities, and system crashes.”

**- Sir Tony Hoare, Turing
Award Recipient (1980)**



Code File Edit Selection View Go Run Terminal Window Help

options.ml — demo

options.ml ×

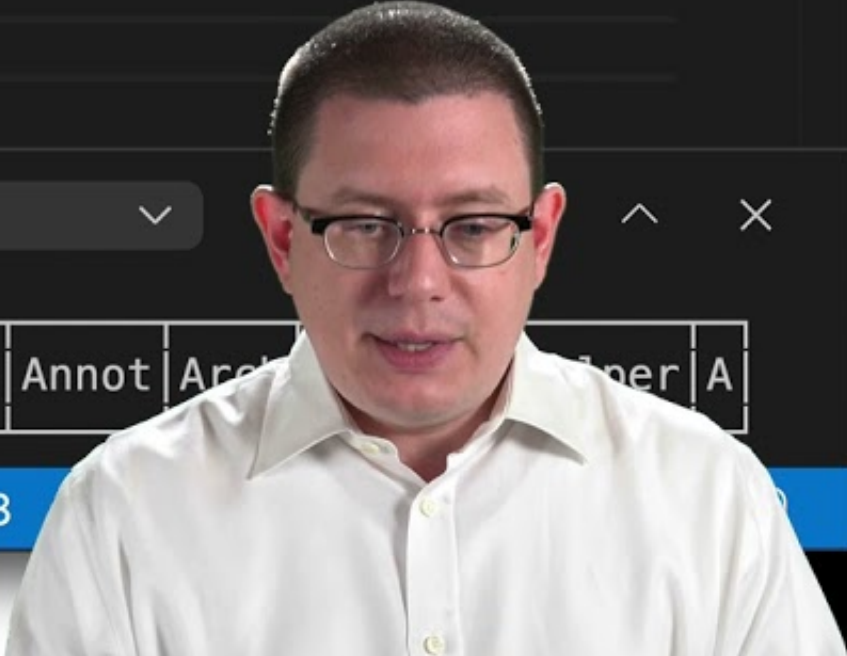
options.ml > [?] list_max

```
1 let get_val default = function
2   | None -> default
3   | Some x -> x
4
5 let rec list_max (lst : 'a list) : 'a option =
6   match lst with
7   | [] -> None
8   | h :: t -> match list_max t with
9     | None -> Some h
10    | Some m -> Some (max h m)
11
```

PROBLEMS TERMINAL ... 1: ocamlrun

Afl_instrument Alias_analysis Allocated_const Annot Arch inner A

<< master* 0 0 Ln 10, Col 31 Spaces: 2 UTF-8



<https://youtu.be/lByolw5wpao>

Question

- How would you implement maximum of a list?

```
let rec max_list (lst : int list) : int =  
  match lst with  
    [] -> ???  
  | h::t -> max(h, max_list(t))
```

Ocaml likes to use Option for such a situation

Type *t option*

- A value *v* has type **t option** if it is either:
 - – the value **None**, or
 - – a value **Some** *v*, and *v* has type *t*
 - type 'a option = None | Some of 'a
- Options can signal there is no useful result to the computation
 - • Example: we loop up a value in a hash table using a key. – If the key is present in the hash table then we return Some *v* where *v* is the associated value
 - – If the key is not present, we return None

Constructing an option

- None
- Some 1
- Some “hi”

Accessing an option

```
match e with  
  None -> ...  
| Some x -> ...
```

Revisit: What is max of empty list?

```
let max (x, y) =  
  if x > y then x else y
```

```
let rec max_list (lst : int list) : int option =  
  match lst with  
    []      -> None  
  | h::t    -> match max_list(t) with  
                 None      -> Some h  
                 | Some x   -> Some (max(h, x))
```

Very stylish!

...no possibility of exceptions

...no chance of programmer ignoring a “null return”

Exercise #1

- Write a function **last** : 'a list -> 'a option that returns the last element of a list.

```
# last ["a" ; "b" ; "c" ; "d"];;  
- : string option = Some "d"  
# last [];;  
- : 'a option = None
```

Exercise #2

- Find the last two elements of a list.

```
# last_two ["a"; "b"; "c"; "d"];;  
- : (string * string) option = Some ("c", "d")  
# last_two ["a"];;  
- : (string * string) option = None
```

Exercise #1

- Write a function **at : int -> 'a list -> 'a option** that returns the n-th element of a list.

```
# at 2 ["a"; "b"; "c"; "d"; "e"];;  
- : string option = Some "c"  
  
# at 2 ["a"];;  
- : string option = None
```