Table of Contents

TP 6 - Programmation pour la préparation à l'agrégation maths option info

TP 6 : Lambda calcul, représentations, calculs et quelques termes pratiques.

- Référence en théorie : ce poly en français de Jean Goubault-Larrecqu (ENS Cachan) (http://www.lsv.fr/%7Egoubault/Lambda/lambda.pdf) ou le livre "Logique réduction résolution", par René Lalement.
- Pour la pratique : ce post de blog en anglais (http://matt.might.net/articles/python-church-y-combinator/)
- Pour plus, cette page wikipédia (https://fr.wikipedia.org/wiki/Lambda-calcul).

- En OCaml.

```
In [1]: let print = Printf.printf;;
        Sys.command "ocaml -version";;
```

```
Out[1]: val print : ('a, out_channel, unit) format → 'a = <fun>

        The OCaml toplevel, version 4.04.2

Out[1]: - : int = 0
```

Représentation des λ-termes en OCaml

Grammaire

Avec une grammaire BNF (https://fr.wikipedia.org/wiki/Forme_de_Backus-Naur), si `<var>` désigne un nom d'expression valide (on se limitera à des noms en miniscules consistitués des 26 lettres a,b,..,z) :

```
<exp> ::= <var>
| <exp>(<exp>)
| fun <var> → <exp>
| (<exp>)
```

Ex1. Ecrire un type Caml représentant les λ-termes

In [3]:
```
type variable = string;;

type terme =
   | V of variable
   | A of terme * terme
   | F of variable * terme
;;
```

Out[3]:
```
type variable = string
```

Out[3]:
```
type terme = V of variable | A of terme * terme | F of variable * terme
```

Par exemple, l'identité est le terme $\lambda x.\, x$.

In [5]:
```
let identite = F("x", V("x"));;
```

Out[5]:
```
val identite : terme = F ("x", V "x")
```

In [7]:
```
let identite_2 = F("y", V("y"));;
```

Out[7]:
```
val identite_2 : terme = F ("y", V "y")
```

Les deux termes sont différents mais égaux à α-renomage près.

Un autre exemple est le terme $\Omega = (\lambda x.\, xx)(\lambda x.\, xx)$ (qui est le plus petit terme dont l'exécution par β-réduction ne termine pas - cf ce poly p7 (http://www.lsv.fr/%7Egoubault/Lambda/lambda.pdf) si besoin).

In [11]:
```
let omega = A(F("x", A(V("x"), V("x"))), F("x", A(V("x"), V("x"))));;
```

Out[11]:
```
val omega : terme = A (F ("x", A (V "x", V "x")), F ("x", A (V "x", V "x")))
```

Ex2. Ecrire une fonction qui affiche un λ-terme en une chaîne de caractère

C'est très rapide.

In [9]:
```
let sprintf = Format.sprintf;;

let rec string_of_terme = function
    | V(s) -> s
    | A(u, v) -> sprintf "(%s)(%s)" (string_of_terme u) (string_of_terme v)
    | F(s, u) -> sprintf "λ %s. (%s)" s (string_of_terme u)
;;
```

```
Out[9]: val sprintf : ('a, unit, string) format → 'a = <fun>
Out[9]: val string_of_terme : terme → variable = <fun>
```

In [12]:
```
print_endline (string_of_terme identite);;
print_endline (string_of_terme identite_2);;
print_endline (string_of_terme omega);;
```

```
λ x. (x)
Out[12]: - : unit = ()
λ y. (y)
Out[12]: - : unit = ()
(λ x. ((x)(x)))(λ x. ((x)(x)))
Out[12]: - : unit = ()
```

Ex3. Bonus : Ecrire une fonction qui transforme un λ-terme en une chaîne représentant une fonction "anonyme" exécutable par Python. Rappel : `lambda x: ...` correspond à $\lambda x. <...>$

In [13]:
```
let sprintf = Format.sprintf;;

let rec python_of_terme = function
    | V(s) -> s
    | A(u, v) -> sprintf "(%s)(%s)" (python_of_terme u) (python_of_terme v)
    | F(s, u) -> sprintf "lambda %s: (%s)" s (python_of_terme u)
;;
```

```
Out[13]: val sprintf : ('a, unit, string) format → 'a = <fun>
Out[13]: val python_of_terme : terme → variable = <fun>
```

In [15]:
```
print_endline (python_of_terme identite);;
print_endline (python_of_terme identite_2);;
print_endline (python_of_terme omega);;
```

```
lambda x: (x)
Out[15]: - : unit = ()
lambda y: (y)
Out[15]: - : unit = ()
(lambda x: ((x)(x)))(lambda x: ((x)(x)))
Out[15]: - : unit = ()
```

On peut ensuite simplement appeler Python sur un terme et vérifier s'il s'exécute ou non.

In [35]:
```
let execute_python_string (s : string) : int =
    Sys.command (sprintf "python -c 'print(%s)'" s)
;;
```

```
Out[35]: val execute_python_string : string → int = <fun>
```

In [36]:
```
execute_python_string (python_of_terme identite);;
```

```
<function <lambda> at 0x7fd84099c5f0>
```

Out[36]:
```
- : int = 0
```

On peut vérifier avec Python que ce terme Ω ne termine pas :

In [34]:
```
execute_python_string (python_of_terme omega);;
```

```
Traceback (most recent call last):
  File "<string>", line 1, in <module>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
```

```
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
```

```
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
```

```
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
```

```
                File "<string>", line 1, in <lambda>
                File "<string>", line 1, in <lambda>
                File "<string>", line 1, in <lambda>
                File "<string>", line 1, in <lambda>
                File "<string>", line 1, in <lambda>
                File "<string>", line 1, in <lambda>
                File "<string>", line 1, in <lambda>
```

```
Out[34]: - : int = 1
```

```
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
```

```
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
```

```
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
```

```
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
```

```
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
```

```
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
```

```
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
```

```
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
File "<string>", line 1, in <lambda>
```

```
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
```

```
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
  File "<string>", line 1, in <lambda>
RuntimeError: maximum recursion depth exceeded
```

Calculs ?

Quelques termes utiles ?

On peut définir des λ-termes utiles avec notre représentation, et on vérifiera ensuite en les exécutant via Python qu'ils sont corrects.

Une valeur "nulle"

In [37]:
```
let none = F("x", V("x"));;
```
Out[37]:
```
val none : terme = F ("x", V "x")
```

Composition

In [54]:
```
let compose u v = A(u, v);;
```
Out[54]:
```
val compose : terme -> terme -> terme = <fun>
```

Conditionnelles

Ce n'est qu'un des encodages possibles.

In [38]:
```
let si = F("cond", F("v", F("f", A(A(V("cond"), V("v")), V("f")))));;
```

```
val si : terme = F ("cond", F ("v", F ("f", A (A (V "cond", V "v"), V "f"))))
```

In [39]:
```
print_endline (string_of_terme si);;
```

```
λ cond. (λ v. (λ f. (((cond)(v))(f))))
```

```
- : unit = ()
```

In [40]:
```
let vrai = F("v", F("f", V("v")));;
let faux = F("v", F("f", V("f")));;
```

```
val vrai : terme = F ("v", F ("f", V "v"))
```

```
val faux : terme = F ("v", F ("f", V "f"))
```

In [41]:
```
print_endline (string_of_terme vrai);;
print_endline (string_of_terme faux);;
```

```
λ v. (λ f. (v))
```

```
- : unit = ()
```

```
λ v. (λ f. (f))
```

```
- : unit = ()
```

Nombres entiers en codage de Chuch

On rappelle que pour $n \in \mathbb{N}$, $[n] = \lambda f.\, \lambda x.\, f^n(x)$ est son codage dit codage de Church dans les λ-termes.

In [45]:
```
let zero = F("f", F("x", V("x")));;
let un = F("f", F("x", A(V("f"), V("x"))));;
```

```
val zero : terme = F ("f", F ("x", V "x"))
```

```
val un : terme = F ("f", F ("x", A (V "f", V "x")))
```

In [50]:
```
let terme_of_int (n : int) : terme =
    let rec aux = function
        | 0 -> V("x")
        | n -> A(V("f"), aux (n-1))
    in
    F("f", F("x", (aux n)))
;;
```

```
val terme_of_int : int → terme = <fun>
```

In [51]:
```
let deux = terme_of_int 2;;
```

```
val deux : terme = F ("f", F ("x", A (V "f", A (V "f", V "x"))))
```

In [53]:
```
print_endline (string_of_terme deux);;
execute_python_string (python_of_terme deux);;
```

```
λ f. (λ x. ((f)((f)(x))))
```

```
- : unit = ()
```

```
<function <lambda> at 0x7f453cab55f0>
```

```
- : int = 0
```

On peut faire l'opération inverse, interpréter un entier de Church en Python.

In [55]:
```
let entier_natif_python = "lambda c: c(lambda x: x+1)(0)";;
```

Out[55]:
```
val entier_natif_python : string = "lambda c: c(lambda x: x+1)(0)"
```

In [56]:
```
execute_python_string (sprintf "(%s)(%s)" entier_natif_python (python_of_terme deux));;
```

```
2
```

Out[56]:
```
- : int = 0
```

> On est obligé de passer par une astuce, parce que $\lambda x.\, x+1$ et 0 ne sont pas des λ-termes.

In [58]:
```
execute_python_string (sprintf "(%s)(%s)" entier_natif_python (python_of_terme (terme_of_int 10)));;
```

```
10
```

Out[58]:
```
- : int = 0
```

Bien sûr, tout ça est très limité !

In [66]:
```
execute_python_string (sprintf "(%s)(%s)" entier_natif_python (python_of_terme (terme_of_int 100)));;
```

```
s_push: parser stack overflow
MemoryError
```

Out[66]:
```
- : int = 1
```

Successeur

Le successeur s'écrit succ $= \lambda n.\, \lambda f.\, \lambda z.\, f(n(f)(z))$.

In [72]:
```
let successeur = F("n", F("f", F("z", A(V("f"), A(A(V("n"), V("f")), V("z"))))));;
```

Out[72]:
```
val successeur : terme =
  F ("n", F ("f", F ("z", A (V "f", A (A (V "n", V "f"), V "z")))))
```

In [92]:
```
print_endline (string_of_terme successeur);;
```

```
λ n. (λ f. (λ z. ((f)(((n)(f))(z)))))
```

Out[92]:
```
- : unit = ()
```

In [70]:
```
let dix = terme_of_int 10;;
let onze = A(successeur, dix);;
```

Out[70]:
```
val dix : terme =
  F ("f",
   F ("x",
    A (V "f",
     A (V "f",
      A (V "f",
       A (V "f",
        A (V "f",
         A (V "f", A (V "f", A (V "f", A (V "f", A (V "f", V "x"))))))))))))
```

Out[70]:
```
val onze : terme =
  A (F ("n", F ("f", F ("z", A (V "f", A (A (V "n", V "f"), V "z"))))),
   F ("f",
    F ("x",
     A (V "f",
      A (V "f",
       A (V "f",
        A (V "f",
         A (V "f",
          A (V "f", A (V "f", A (V "f", A (V "f", A (V "f", V "x"))))))))))))
```

A noter que ce terme `onze` ne sera pas le même que celui (plus court) obtenu par `terme_of_int 11` :

In [73]:
```
let onze2 = terme_of_int 11;;
```

Out[73]:
```
val onze2 : terme =
  F ("f",
   F ("x",
    A (V "f",
     A (V "f",
      A (V "f",
       A (V "f",
        A (V "f",
         A (V "f", A (V "f", A (V "f", A (V "f", A (V "f", V "x"))))))))))))
```

Mais ils s'exécutent de la même façon :

In [74]:
```
execute_python_string (sprintf "(%s)(%s)" entier_natif_python (python_of_terme dix));;
execute_python_string (sprintf "(%s)(%s)" entier_natif_python (python_of_terme onze));;
execute_python_string (sprintf "(%s)(%s)" entier_natif_python (python_of_terme onze2));;
```

```
10
```

Out[74]:
```
- : int = 0
```

```
11
```

Out[74]:
```
- : int = 0
```

```
11
```

Out[74]:
```
- : int = 0
```

Addition

La somme s'écrit somme $= \lambda n.\, \lambda m.\, \lambda f.\, \lambda z.\, n(f)(m(f)(z))$.

In [90]:
```
let somme = F("n", F("m", F("f", F("z", A((A(V("n"), V("f"))), A((A(V("m"), V("f"))), V("z")) )))));;
```

Out[90]:
```
val somme : terme =
  F ("n",
   F ("m",
    F ("f", F ("z", A (A (V "n", V "f"), A (A (V "m", V "f"), V "z"))))))
```

In [91]:
```
print_endline (string_of_terme somme);;
```

```
λ n. (λ m. (λ f. (λ z. (((n)(f))(((m)(f))(z))))))
```

Out[91]:
```
- : unit = ()
```

In [87]:
```
let trois = A(A(somme, un), deux);;
```

Out[87]:
```
val trois : terme =
  A
   (A
     (F ("n",
       F ("m",
        F ("f", F ("z", A (A (V "n", V "f"), A (A (V "m", V "f"), V "z"))))))),
     F ("f", F ("x", A (V "f", V "x")))),
    F ("f", F ("x", A (V "f", A (V "f", V "x")))))
```

Comme pour le successeur, ce terme est bien plus compliqué que l'encodage de Church, mais ils s'exécutent de la même manière.

In [88]:
```
let trois2 = terme_of_int 3;;
```

Out[88]:
```
val trois2 : terme = F ("f", F ("x", A (V "f", A (V "f", A (V "f", V "x")))))
```

In [89]:
```
execute_python_string (sprintf "(%s)(%s)" entier_natif_python (python_of_terme trois));;
execute_python_string (sprintf "(%s)(%s)" entier_natif_python (python_of_terme trois2));;
```

```
3
```

Out[89]:
```
- : int = 0
```

```
3
```

Out[89]:
```
- : int = 0
```

Multiplication

La multiplication est $\mathrm{mul} = \lambda n.\, \lambda m.\, \lambda f.\, \lambda z.\, m(n(f))(z)$.

In [122]:
```
let mul = F("n", F("m", F("f", F("z", A(A(V("m"), A(V("n"), V("f"))), V("z"))))));;
```

Out[122]:
```
val mul : terme =
  F ("n", F ("m", F ("f", F ("z", A (A (V "m", A (V "n", V "f")), V "z")))))
```

In [123]:
```
let trois = terme_of_int 3 ;;
let six = A(A(mul, trois), deux);;
let six2 = A(A(mul, deux), trois);;
```

Out[123]:
```
val trois : terme = F ("f", F ("x", A (V "f", A (V "f", A (V "f", V "x")))))
```

Out[123]:
```
val six : terme =
  A
   (A
     (F ("n",
       F ("m", F ("f", F ("z", A (A (V "m", A (V "n", V "f")), V "z"))))),
     F ("f", F ("x", A (V "f", A (V "f", A (V "f", V "x")))))),
    F ("f", F ("x", A (V "f", A (V "f", V "x")))))
```

Out[123]:
```
val six2 : terme =
  A
   (A
     (F ("n",
       F ("m", F ("f", F ("z", A (A (V "m", A (V "n", V "f")), V "z"))))),
     F ("f", F ("x", A (V "f", A (V "f", V "x")))))),
    F ("f", F ("x", A (V "f", A (V "f", A (V "f", V "x"))))))
```

Comme pour le successeur, ce terme est bien plus compliqué que l'encodage de Church, mais ils s'exécutent de la même manière.

In [124]:
```
execute_python_string (sprintf "(%s)(%s)" entier_natif_python (python_of_terme six));;
execute_python_string (sprintf "(%s)(%s)" entier_natif_python (python_of_terme six2));;
execute_python_string (sprintf "(%s)(%s)" entier_natif_python (python_of_terme (terme_of_int 6)));;
```

```
6
```

Out[124]:
```
- : int = 0
```

```
6
```

Out[124]:
```
- : int = 0
```

```
6
```

Out[124]:
```
- : int = 0
```

Paires

La représentation de la paire est simplement pair $= \lambda a.\, \lambda b.\, \lambda f.\, f(a)(b)$.

In [127]:
```
let a = V("a") and b = V("b") and f = V("f");;
let pair = F("a", F("b", F("f", A(A(f, a), b))));;
```

Out[127]:
```
val a : terme = V "a"
val b : terme = V "b"
val f : terme = V "f"
```

Out[127]:
```
val pair : terme = F ("a", F ("b", F ("f", A (A (V "f", V "a"), V "b"))))
```

Et ensuite les deux extracteurs sont immédiats : gauche $= \lambda p.\, p(\lambda a.\, \lambda b.\, a)$ et droite $= \lambda p.\, p(\lambda a.\, \lambda b.\, b)$. (on retrouve `vrai` et `faux`)

In [130]:
```
let gauche = F("f", A(f, vrai));;
let droite = F("f", A(f, faux));;
```

Out[130]:
```
val gauche : terme = F ("f", A (V "f", F ("v", F ("f", V "v"))))
```

Out[130]:
```
val droite : terme = F ("f", A (V "f", F ("v", F ("f", V "f"))))
```

In [133]:
```
let exemple_pair = A(A(pair, deux), trois);;
```

Out[133]:
```
val exemple_pair : terme =
  A
   (A (F ("a", F ("b", F ("f", A (A (V "f", V "a"), V "b")))),
     F ("f", F ("x", A (V "f", A (V "f", V "x"))))),
   F ("f", F ("x", A (V "f", A (V "f", A (V "f", V "x"))))))
```

On vérifie qu'on peut extraire `[2]` et `[3]` de cette paire `[(2, 3)]` :

In [134]:
```
execute_python_string (sprintf "(%s)(%s)" entier_natif_python (python_of_terme (A(gauche, exemple_pair))));;
execute_python_string (sprintf "(%s)(%s)" entier_natif_python (python_of_terme (A(droite, exemple_pair))));;
```

```
2
```

Out[134]:
```
- : int = 0
```

```
3
```

Out[134]:
```
- : int = 0
```

Bonus : prédecesseur

> Avec les paires, c'est possible. Idée de l'algorithme : en ayant `[n]`, on commence par la paire `[(0, 0)]` et on itère la fonction `fun (a,b) → (a+1, a)` exactement `n` fois (et ça c'est facile par définition du codage `[n]`, ce qui donne la paire `[(n, n-1)]` et en récupérant la deuxième coordonnée on a `[n-1]`. C'est corrigé en Exercice 12 du poly de Jean Goubault-Larrecqu.

On va découper ça en morceau :

In [137]:
```
let constructeur_pair u v = A(A(pair, u), v);;
let pi1 u = A(gauche, u);;
let pi2 u = A(droite, u);;
let constructeur_succ u = A(successeur, u);;

let pair_00 = constructeur_pair zero zero;;
```

```
Out[137]: val constructeur_pair : terme → terme → terme = <fun>

Out[137]: val pi1 : terme → terme = <fun>

Out[137]: val pi2 : terme → terme = <fun>

Out[137]: val constructeur_succ : terme → terme = <fun>

Out[137]: val pair_00 : terme =
            A
             (A (F ("a", F ("b", F ("f", A (A (V "f", V "a"), V "b")))),
               F ("f", F ("x", V "x"))),
             F ("f", F ("x", V "x")))
```

In [139]:
```
let p = V("p");;
let succ_1 = F("p", constructeur_pair (constructeur_succ(pi1(p))) (pi1(p)));;
```

```
Out[139]: val p : terme = V "p"

Out[139]: val succ_1 : terme =
            F ("p",
             A
              (A (F ("a", F ("b", F ("f", A (A (V "f", V "a"), V "b")))),
                A (F ("n", F ("f", F ("z", A (V "f", A (A (V "n", V "f"), V "z"))))),
                 A (F ("f", A (V "f", F ("v", F ("f", V "v")))), V "p"))),
              A (F ("f", A (V "f", F ("v", F ("f", V "v")))), V "p")))
```

In [140]:
```
let n = V("n");;
let predecesseur = F("n", pi2(A(A(n, succ_1), pair_00)));;
```

```
Out[140]: val n : terme = V "n"

Out[140]: val predecesseur : terme =
            F ("n",
             A (F ("f", A (V "f", F ("v", F ("f", V "f")))),
              A
               (A (V "n",
                 F ("p",
                  A
                   (A (F ("a", F ("b", F ("f", A (A (V "f", V "a"), V "b")))),
                     A
                      (F ("n",
                        F ("f", F ("z", A (V "f", A (A (V "n", V "f"), V "z"))))),
                       A (F ("f", A (V "f", F ("v", F ("f", V "v")))), V "p"))),
                    A (F ("f", A (V "f", F ("v", F ("f", V "v")))), V "p")))),
                A
                 (A (F ("a", F ("b", F ("f", A (A (V "f", V "a"), V "b")))),
                   F ("f", F ("x", V "x"))),
                  F ("f", F ("x", V "x"))))))
```

In [142]:
```
let cinq = A(predecesseur, (terme_of_int 6));;
let cinq2 = terme_of_int 5;;
```

Out[142]:
```
val cinq : terme =
  A
   (F ("n",
     A (F ("f", A (V "f", F ("v", F ("f", V "f")))),
      A
       (A (V "n",
         F ("p",
          A
           (A (F ("a", F ("b", F ("f", A (A (V "f", V "a"), V "b")))),
             A
              (F ("n",
                F ("f", F ("z", A (V "f", A (A (V "n", V "f"), V "z"))))),
               A (F ("f", A (V "f", F ("v", F ("f", V "v")))), V "p"))),
            A (F ("f", A (V "f", F ("v", F ("f", V "v")))), V "p")))),
      A
       (A (F ("a", F ("b", F ("f", A (A (V "f", V "a"), V "b")))),
         F ("f", F ("x", V "x"))),
        F ("f", F ("x", V "x"))))))),
    F ("f",
     F ("x",
      A (V "f", A (V "f", A (V "f", A (V "f", A (V "f", A (V "f", V "x")))))))))
```

Out[142]:
```
val cinq2 : terme =
  F ("f",
   F ("x", A (V "f", A (V "f", A (V "f", A (V "f", A (V "f", V "x")))))))
```

Comme pour le successeur, ce terme est bien plus compliqué que l'encodage de Church, mais ils s'exécutent de la même manière.

In [143]:
```
execute_python_string (sprintf "(%s)(%s)" entier_natif_python (python_of_terme cinq));;
execute_python_string (sprintf "(%s)(%s)" entier_natif_python (python_of_terme cinq2));;
```

```
5
```

Out[143]:
```
- : int = 0
```

```
5
```

Out[143]:
```
- : int = 0
```

On vérifie que `pred 0 = 0` :

In [146]:
```
let zero2 = A(predecesseur, zero);;
```

Out[146]:
```
val zero2 : terme =
  A
   (F ("n",
     A (F ("f", A (V "f", F ("v", F ("f", V "f")))),
      A
       (A (V "n",
         F ("p",
          A
           (A (F ("a", F ("b", F ("f", A (A (V "f", V "a"), V "b")))),
             A
              (F ("n",
                F ("f", F ("z", A (V "f", A (A (V "n", V "f"), V "z"))))),
               A (F ("f", A (V "f", F ("v", F ("f", V "v")))), V "p"))),
            A (F ("f", A (V "f", F ("v", F ("f", V "v")))), V "p")))),
      A
       (A (F ("a", F ("b", F ("f", A (A (V "f", V "a"), V "b")))),
         F ("f", F ("x", V "x"))),
        F ("f", F ("x", V "x"))))))),
    F ("f", F ("x", V "x")))
```

```
In [147]: execute_python_string (sprintf "(%s)(%s)" entier_natif_python (python_of_terme zero2));;
          execute_python_string (sprintf "(%s)(%s)" entier_natif_python (python_of_terme zero));;
```

```
0
Out[147]: - : int = 0
          0
Out[147]: - : int = 0
```

Listes

Récursion U

Point fixe Y

Bonus : la factorielle en λ-calcul

Conclusion

Fin. À la séance prochaine. Le TP6 traitera de ?? (en février).