

Programmation fonctionnelle avancée

Le cours de prog. fonctionnelle avancée s'intéresse tout d'abord à...

Vrai but : chasser les trolls.

Troll : tout impératif ! tout fonctionnel !

Futé : problème $\leadsto$ « bonne » structure $\leadsto$ style adapté

Fonctionnel ?

- Tout est valeur (expressions, fonctions)
- Persistance (jamais de modif)

Prog. déclarative $\leadsto$ abstraction, haut niveau pourquoi $\neq$ comment

Ici : représentation de collections

Programmation fonctionnelle avancée

Haskell, ML, Erlang, Scheme, Lisp...

(Python), Rust, Javascript

OCAML

- Fonctionnel + traits impératifs
- Fortement typé
- Évaluation stricte
- Compilé $\leadsto$ vers natif, efficace
- Interprété $\leadsto$ vers bytecode, générique

garanties par construction

Persistence $\implies$ gestion mémoire efficace, très complexe, cachée

Efficace ?

Pénible ?

Utilisé ?

100% pur jus ?

Efficace ?

Compilo natif + bytecode léger

Pénible ?

Utilisé ?

100% pur jus ?

Efficace ?

Compilo natif + bytecode léger

Pénible ?

Déclaratif $\leadsto$ proche des spéc.

Utilisé ?

100% pur jus ?

Efficace ?

Compilo natif + bytecode léger

Pénible ?

Déclaratif $\leadsto$ proche des spéc.

Utilisé ?

META, JaneStreet, VPNkit Docker, blockchains Tezos...

100% pur jus ?

Efficace ?

Compilo natif + bytecode léger

Pénible ?

Déclaratif $\leadsto$ proche des spéc.

Utilisé ?

META, JaneStreet, VPNkit Docker, blockchains Tezos...

100% pur jus ?

NON c'est ce cours !

Débroussaillage

Parties standard : propriétés $\leftrightarrow$ **structure** + **outils**

Oui mais *persistantes* : partage en mémoire. . .

Mélange impératif/fonctionnel $\leadsto$ partage maximal + comparaison φ

Hash-consing basique puis plus évolué

Gains en abstraction. . .

—

TP pour manipuler, TD pour les aspects plus théoriques