PF
Programmation Fonctionnelle

Legond-Aubry Fabrice

fabrice.legond-aubry@parisnanterre.fr

Plan du Cours

Programmation Fonctionnelle – WTF ?

Rappels sur les Génériques (et autres)

Fonctions

Lambda Calculs

Streams

Compléments

Définitions (Hors Langage)

Exemple

Plan du Cours

Programmation Fonctionnelle – WTF ?

Rappels sur les Génériques (et autres)

Fonctions

Lambda Calculs

Streams

Compléments

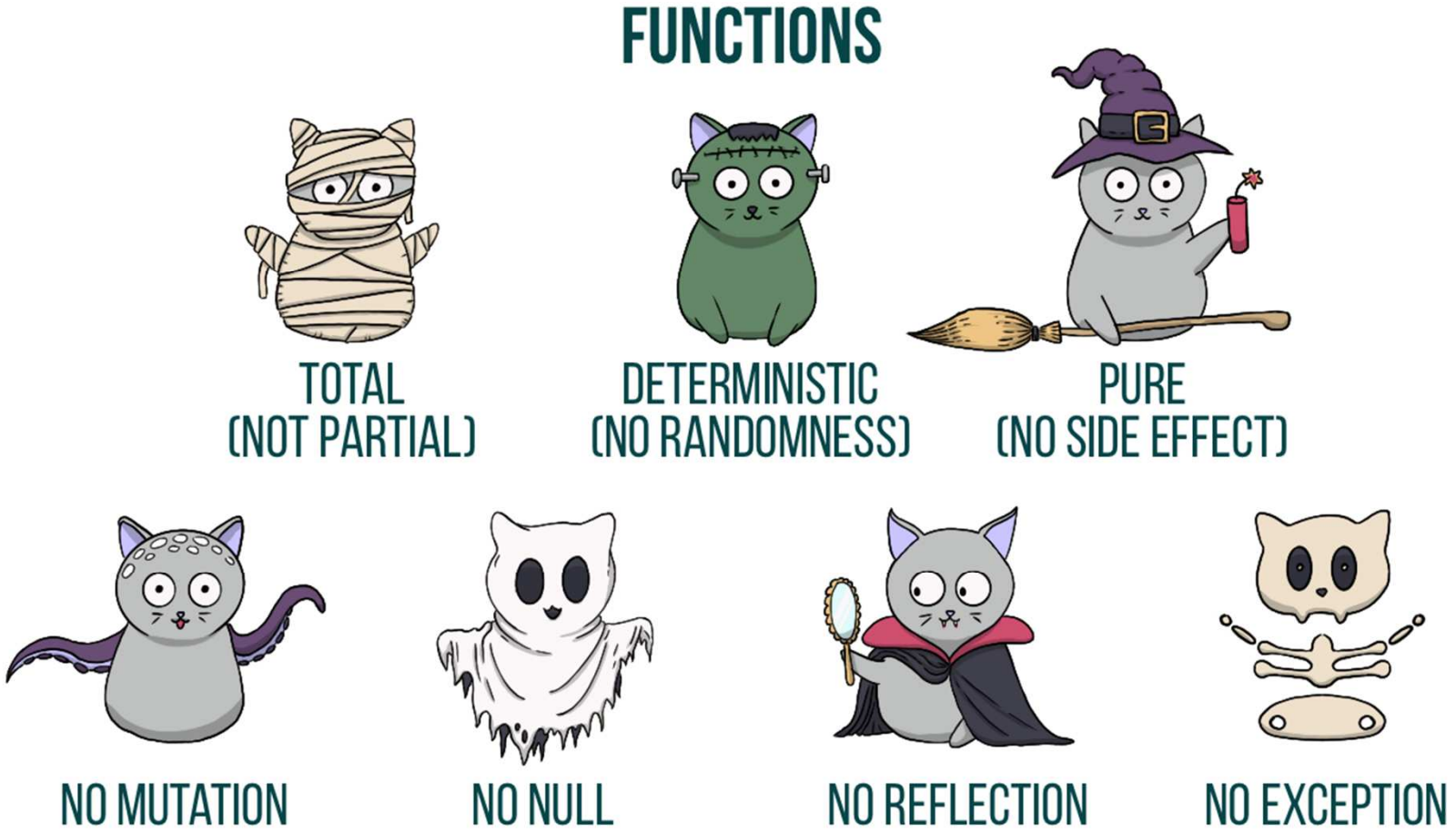
Définitions (Hors Langage)

Exemple

Programmation Fonctionnelle – WTF ?

- En informatique, la **programmation fonctionnelle** est un paradigme de programmation où les programmes sont codés en appliquant et en composant (au sens mathématique) des fonctions.
- C'est un paradigme déclaratif dans lesquels les langages définissent des arbres d'expressions qui retournent chacune une valeur plutôt qu'une séquence d'instructions (impératives) qui change l'état du programme.
- Il y a peu de langage totalement « paradigme PF » car il impose des contraintes fortes.
  - ✓ Certains diront: « existe-t-il vraiment un langage fonctionnel pure ? »
  - ✓ Haskell en un exemple de PF
- Il y a souvent des hybridations programmation impérative / programmation fonctionnelle
  - ✓ Java en est un exemple, Kotlin et Scala en sont d'autres

Programmation Fonctionnelle – WTF ?



Programmation Fonctionnelle – WTF ?

« Total »

Retourne une valeur en sortie (output) pour chaque valeur en entrée.

Chaque valeur **possible** d'entrée doit avoir une sortie

« Deterministic »

Retourne toujours les **mêmes** valeurs en sortie pour les **mêmes** entrées quelque soit le nombre de fois où le code est exécuté.
Peut dépendre d'une donnée extérieure

« Pure »

Une fonction pure est une fonction qui qui ne calcul son résultat qu'en fonction de la donnée.

Le seul effet du code est de calculer une valeur (de sortie).

IL N'Y A PAS DE NOTION DE CHANGEMENT D'ETAT.

« Pure » est un sous ensemble de « deterministic »

Le retour de la fonction ne peut donc pas dépendre d'une information cachée, d'un état QUI CHANGE, d'une entrée/sortie (I/O)

Programmation Fonctionnelle – WTF ?

	FP	POO
Définition	Évaluation de compositions de fonctions	Concepts objets, changement d'état
Données	Données immutables	Données mutables (ou non)
Modèle	Déclaratif	Instructions (modèle impératif)
Parallélisation	Parallélisation naturelle	Non native (algorithme à trouver)
Exécution	Exécutions non ordonnées	Ordre d'exécution important
Itération	Récursion pour le parcours des données	Boucle, récursion
Eléments	Fonctions, variables (math)	Objets, méthodes, variables (info)

Programmation Fonctionnelle – WTF ?

Programmation Fonctionnelle VS POO

- « Avantages » / « Inconvénients » PF
  - ✓ + Utilisation de fonctions dont le comportement est prédictible, sans effet de bords
  - ✓ + Retourne exactement ce qui est attendu
  - ✓ + Se concentre sur un typage fort, focus sur l'efficacité et l'optimisation
  - ✓ - Paradigme peu connu, peu usité
  - ✓ - moins de documentation, moins de base de code
  - ✓ - génération de nombreuses petites fonctions
  - ✓ + Plus adapté pour traiter de grande quantité de données
- Caractéristiques :
  - ✓ Pas d'état, donc facilement parallélisable
  - ✓ travail sur des données "fixes" (une source)

Programmation Fonctionnelle – WTF ?

Programmation Fonctionnelle VS POO

- « Avantages » / « Inconvénients » POO
  - ✓ + Très connu et documenté. Le paradigme est clair et compréhensible.
  - ✓ + “Moins Mathématique”. Utilise un style impératif. Suites d’instructions comme un livre à lire
  - ✓ - Des comportements non prédictible ou non anticipé
  - ✓ - Parallèlisation complexe (threads, mutex, synchronisation, deadlock) pour les données partagées
  - ✓ - Parallèlisation peut générer un surcoût important (perte de l’accélération linéaire)
  - ✓ - Peut générer des effets de bords avec un impact sur les performances
- Caractéristiques :
  - ✓ Gestion de l’état
  - ✓ Représentation simple du problème à traiter (“Mimer” le réel)
  - ✓ Partage / Copie / Mute des données