UTC503 : Paradigmes de programmation
—
UTC503 : Paradigmes de programmation — Introduction

Xavier Crégut
<Prénom.Nom@enseeiht.fr>

ENSEEIHT
Sciences du Numérique

Avant de commencer...

Question : Que vous évoquent les termes :

1. programmation impérative
2. programmation fonctionnelle
3. programmation déclarative (en particulier programmation logique)
4. programmation objet
5. programamtion réactive

Objectifs du module

Objectifs : Connaître et approfondir les principaux paradigmes de programmation : impératif, logique, fonctionnel, réactif, objet ; savoir les mettre en œuvre ; comprendre leurs différences.

Compétences visées : Pouvoir aborder un nouveau langage de programmation ou une nouvelle bibliothèque en reconnaissant les usages dans ceux-ci des principaux paradigmes. La plupart des langages de programmation actuels étant hybrides, et s'ouvrant de plus en plus au paradigme fonctionnel, les connaissances dans un paradigme seront utilisables au-delà de celui-ci.

Pré-requis : Connaître un langage de programmation avancé, comme java, et maîtriser les notions d'algorithme, de procédure et fonction, d'objet, de méthode, et d'héritage.

Organisation des séances

Où : ENSEEIHT, 2 rue Camichel, salle C201 (salle machines Linux)

- Entrée possible via l'accueil (jusqu'à 18h30, éventuellement 19h).
- Sortie libre par les tourniquets (ne pas prendre celui qui mène au parking !)
- ~~Des badges d'accès seront (peut-être) founis via l'IPST-CNAM...~~

Quand : lundi de 18h à 21h

- du 11 octobre 2021 au 29 octobre 2021
- pas de cours le 1er novembre 2021 ;-)

Séance type :

1. Retour sur les exercices précédents. Questions/réponses.
2. Présentation du nouveau cours (succinct)
3. Exercices
4. Avec une petite pause pour souffler un peu...

Examen Session 1 :

- 2h heures, **examen sur papier**, notes de cours autorisées...
- le **6 décembre 2021**

Examen Session 2 : ???

Des questions ?

Planning prévisionnel

1. Algorithmique impérative en Python
2. Les sous-programmes en Python (vers du fonctionnel)
3. Structures de données (conteneurs) en Python
4. Programmation Objet (Python)
5. Programmation fonctionnelle (Python et Caml)
6. Programmation logique (Prolog)
7. Examen

Différents paradigmes de programmation

Paradigme fonctionnel

```
1  Pgcd(a, b) =
2      Si a = b Alors
3          a
4      SinonSi a > b Alors
5          pgcd(a-b, b)
6      Sinon
7          pgcd(a, b-a)
8      FinSi
```

Paradigme impératif

```
1  Pgcd(a, b) =
2      TantQue a <> b Faire
3          Si a > b Alors
4              a <- a - b
5          Sinon
6              b <- b - a
7          FinSi
8      FinTQ
9      Résultat <- a
```

Paradigme déclaratif (logique)

```
1  Pgcd(A, A, A).
2  Pgcd(A, B, R) :- A > B, A1 is A - B, Pgcd(A1, B, R).
3  Pgcd(A, B, R) :- A < B, Pgcd(B, A, R).
```

Comparaison des différents paradigmes

Paradigme impératif

- Le programme est une instruction complexe.
- Un programme est composé d'instructions qui modifient l'état du programme
- Ces effets de bord rendent difficile le raisonnement sur un programme, la parallélisation du code (problèmes de synchronisation), etc.

Paradigme fonctionnel

- Le programme est une fonction.
- Un programme est composé de fonctions (au sens mathématique).
- Pas d'état. Pas d'effet de bord.
  - Raisonnement facilité sur le programme.
  - Mise en œuvre plus facile (voire gratuite) de la concurrence.

Paradigme déclaratif

- Le programme est un but.
- Un programme est composé d'un ensemble de faits et règles.
- Le programmeur ne donne pas de description opératoire (l'ordre d'application des règles)
  - Concurrence gratuite
  - Raisonnement facilité

Évolution : De plus en plus, les langages sont multi-paradigmes !

Quelques enjeux de la programmation

Faciliter d'écriture d'un programme

- de petite taille (quelques centaines à milliers de lignes), de grande taille (millions de lignes)
- par un informaticien, par un fonctionnel ?

Capacité à faire évoluer un programme

- prendre en compte de nouveaux besoins
- prendre en compte les évolutions réglementaires, technologiques, etc.
- identifier et corriger les erreurs...

Capacité à raisonner sur un programme (pour gagner en confiance dans le programme)

- Est-ce que le programme fait ce qui est attendu ?
- Preuve ? Vérification exhaustive ? Test ? Certification ?

Avoir des programmes efficients : bien utiliser les ressources disponibles

- multi-coeurs, multi-processeurs, architectures parallèles, etc.
- espace mémoire, communications réseau, etc.

Quels sont les enjeux prioritaires ?