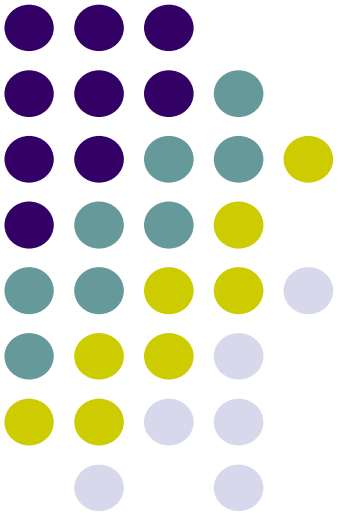
MI-GLSD-M1 -UEM213 : Paradigmes de langages de Programmation

Chapitre I: Introduction et Concepts de Base

A. HARICHE
Université de Djilali Bounaama, Khemis Meliana (UDBKM)
Faculté des sciences et de la technologie
Département de mathématiques et d'informatique

a.hariche@univ-dbkm.dz



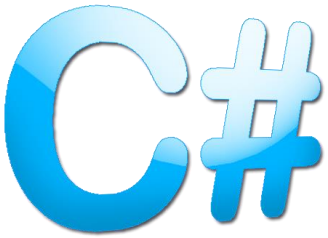
Transparents inspirés par Christian Schulte et P. Van Roy et Seif Haridi

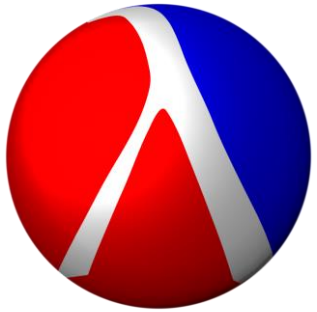
Paradigmes des langages de programmation

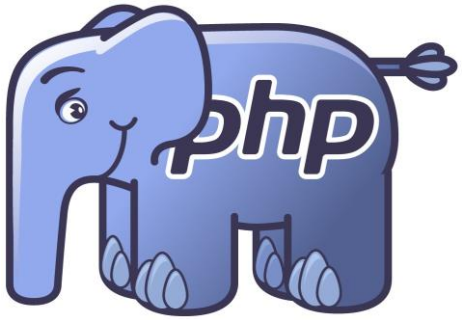
- Inspiré de Louv1.1x et Louv1.2x forment une séquence à deux cours
  - Ensemble, ils enseignent la programmation comme une discipline unifiée qui couvre tous les langages de programmation
  - Nécessite une certaine expérience en programmation et en mathématiques (ensembles, listes, fonctions)
- Les deux cours couvrent quatre thèmes importants :
  - *Programmation fonctionnelle (et structures de données de base)* } Louv1.1x
  - *Sémantique formelle (et complexité de calcul)* } Louv1.1x
  - Abstraction de données (et programmation orientée objet) } Louv1.2x
  - *Concurrence (et flux de données déterministe)* } Louv1.2x
  - Voyons comment cela fonctionne en pratique .

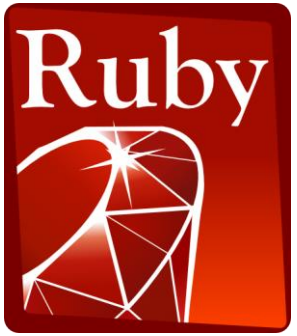
A.HARICHE a.hariche@univ-dbkm.dz

Des centaines de langages de programmation sont utilisés...



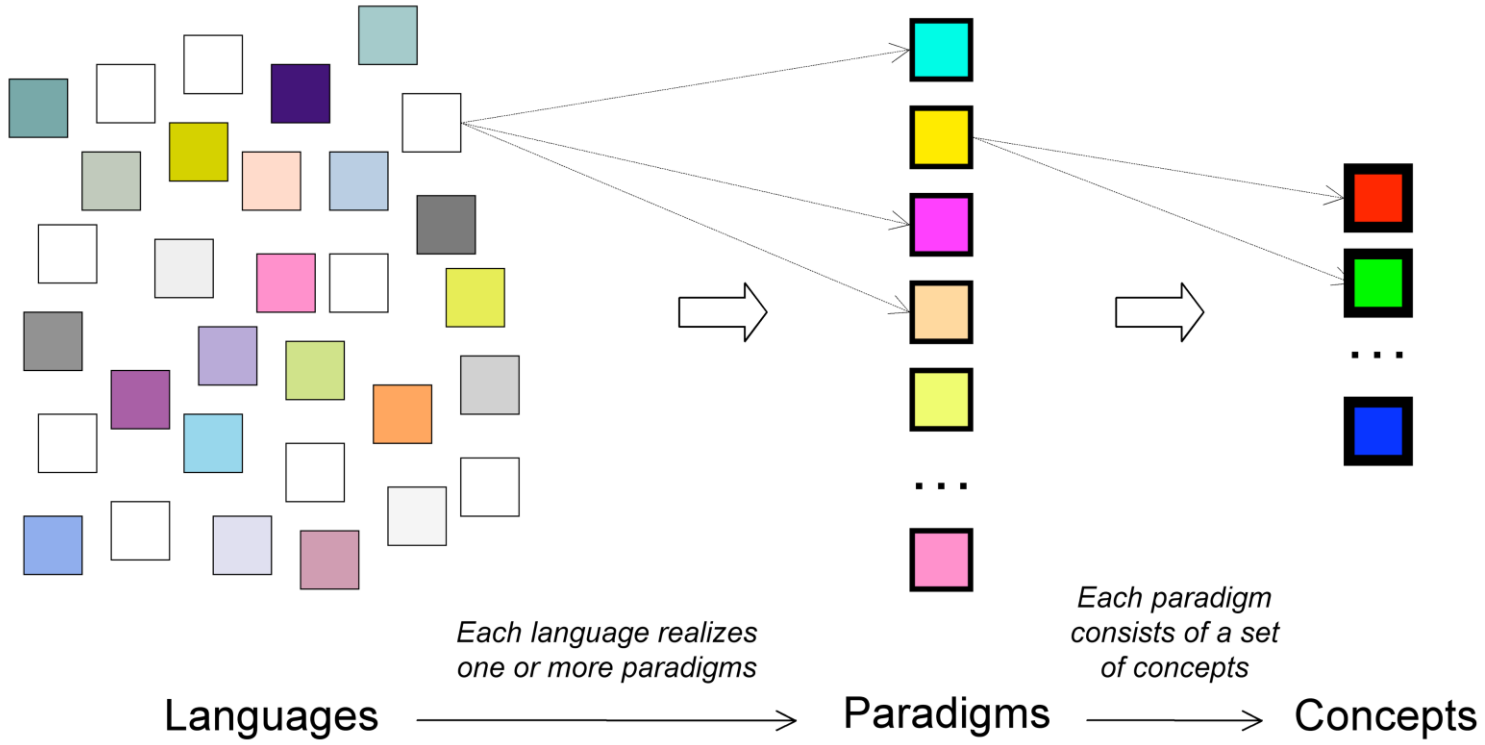






A.HARICHE a.hariche@univ-dbkm.dz

Tellement nombreux, comment pouvons-nous les comprendre ?



- Idée clé : les langages sont basés sur des paradigmes, et il y a beaucoup moins de paradigmes que de langages
- Nous pouvons comprendre plusieurs langages en apprenant peu de paradigmes !

Définition

- Un paradigme de programmation est une approche de la programmation en informatique basée sur un ensemble cohérent de principes ou une théorie mathématique
- Un programme est crée pour résoudre des problèmes
- Tout programme réaliste doit résoudre différents types de problèmes
- Chaque type de problème a besoin de son propre paradigme
- Plusieurs paradigmes et nous devons les combiner dans le même programme

Comment étudier plusieurs paradigmes ?

- Comment étudier des paradigmes correctement (puisque la plupart des langages ne supportent qu'un, ou parfois deux paradigmes) ?
- Chaque langage a sa propre *syntaxe*, sa propre *sémantique*, son propre *système* et ses propres *astuces*.
- Choisissons trois langages, comme Java, Erlang et Haskell, et structurer durant un cours.
- Résultat un cours compliqué et non objectif.
- Solution pragmatique : utiliser un seul langage, Oz, un langage de recherche conçu pour de nombreux paradigmes.
- Concentrer sur les vrais problèmes
- Reference '*Concepts, Techniques, and Models of Computer Programming*', en utilisant Oz pour etudier plusieurs paradigmes

A.HARICHE a.hariche@univ-dbkm.dz

Comment combiner des paradigmes dans un programme ?

- Chaque paradigme est une façon différente de penser
- Comment combiner différentes manière-pensés dans un programme?
- Utiliser le concept d'un langage noyau
- Chaque paradigme a un langage de base simple, son langage noyau, qui contient ses concepts essentiels
- Chaque langage pratique, même s'il est compliqué, peut être traduit facilement dans son langage noyau
- Les différents paradigmes ont des langages de noyau en commun ; souvent il n'y a qu'une seule différence de concept
- Notre premier paradigme avec son langage noyau est la programmation fonctionnelle
- On ajoute ensuite les concepts un par un pour obtenir aux autres paradigmes

A.HARICHE a.hariche@univ-dbkm.dz

Notre approche en bref

- Centaines de langages sont utilisées dans la pratique : on ne peut pas toutes les étudier en un cours ou même en une vie
- Solution : **se concentrer sur les paradigmes**, puisque chaque langage est basé sur un paradigme et alors moins de paradigmes à étudier.
- Un langage par paradigme, c'est trop pour étudier dans un cours, puisque chaque langage est déjà compliquée
- Solution : utiliser **un langage de recherche**, **Oz**, qui peut exprimer de nombreux paradigmes
- Les programmes complexes (réels) doivent combiner des paradigmes, mais comment le faire puisque chaque paradigme est une manière-pensée ?
- Solution : **définir des paradigmes à l'aide des langages noyau**, car différents paradigmes ont beaucoup en commun
- Les langages noyau nous permettent de définir de nombreux paradigmes en nous concentrant sur leurs différences, ce qui est beaucoup plus économique en temps et en efforts

Commençons!!

- Paradigme Fameux en informatique est P.O.O
- La programmation orientée objet P.O.O, avec ses principes cohérents, est clairement un paradigme important
- Et alors quoi dire pour les autres paradigmes?
- La P.O.O n'est-elle pas à ce moment le paradigme le plus important et le plus utile ?
- Réellement, Alors que NON!
- De nombreux autres paradigmes sont extrêmement utiles, souvent plus que la POO ! Par exemple, pour créer des programmes distribués robustes et efficaces sur Internet, la POO ne résout tout simplement pas les bons problèmes.

  La programmation de flux de données multi-agents est la meilleure.
- Cinq paradigmes qui résolvent de nombreux problèmes

Cinq paradigmes

- Cinq paradigmes principale à étudier:
- Programmation impérative
- Programmation fonctionnelle
- Programmation orientée objets
- Programmation logique
- Programmation Concurrente y compris :

Flux de données déterministe, Multi-agent et par des objets Actives

- Ce sont probablement les paradigmes de programmation les plus importants pour une utilisation générale
- Mais il existe bien d'autres paradigmes, faits pour d'autres problèmes : ce cours vous donnent une bonne base pour les étudier plus tard si vous le souhaitez

A.HARICHE a.hariche@univ-dbkm.dz

Idées à apprendre

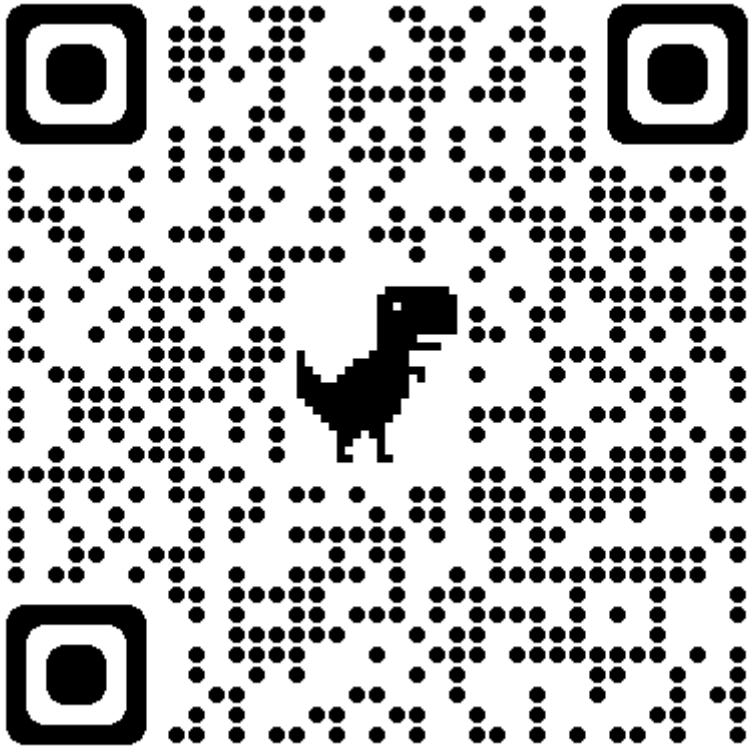
Louv1.1x

- Identifiants et environnements
- Programmation fonctionnelle
- Récursivité
- Programmation invariante
- Listes, arborescences et enregistrements
- Programmation symbolique
- Instanciation
- Généricité
- Programmation d'ordre supérieur
- Complexité et notation Big-O
- Loi de Moore
- Problèmes NP et NP-complets
- Langages du noyau
- Machines abstraites
- Sémantique mathématique

Louv1.2x

- État explicite
- Abstraction des données
- Types de données abstraites et objets
- Polymorphisme
- Héritage
- Héritage multiple
- Programmation orientée objet
- Gestion des exceptions
- Concurrence
- Non-déterminisme
- Ordonnancement et équité
- Synchronisation des flux de données
- Flux de données déterministe
- Agents et flux
- Programmation multi-agents

Lien



https://drive.google.com/drive/folders/1YBCIZzAldeiT19DIfDiREQwP-NAQ1qMN