

Informatique

Informatique 3 : Algorithmique et Programmation

IDENTIFICATION

CODE : BS-3-S2-EC-BMINFO3
ECTS : 3.0

HORAIRES

Cours : 16.0 h
TD : 22.0 h
TP : 0.0 h
Projet : 0.0 h
Face à face
pédagogique : 38.0 h
Travail personnel : 37.0 h
Total : 75.0 h

ÉVALUATION

SUPPORTS PÉDAGOGIQUES

LANGUE D'ENSEIGNEMENT

Anglais

CONTACT

MME BERNARD KNIBBE
Carole
carole.knibbe@insa-lyon.fr

OBJECTIFS RECHERCHÉS PAR CET ENSEIGNEMENT

Cet EC contribue aux compétences ci-dessous [niveau] avec les capacités associées :

- A1. Analyser un système [réel ou virtuel] [niveau 2]
 - Evaluer la complexité d'un algorithme simple
- A2. Exploiter un modèle d'un système réel ou virtuel [niveau 2]
 - Prédire l'évolution des valeurs des variables d'un algorithme donné
- A4. Concevoir un système répondant à un cahier des charges [niveau 2]
 - Concevoir un algorithme pour résoudre un problème et l'implémenter en Python
- C12. Automatiser le traitement et l'extraction de connaissances à partir de données biologiques. [niveau 2]
 - Implémenter un algorithme en Python dans le style impératif
 - Passer un programme du style impératif au style orienté-objet
 - Choisir judicieusement les structures de données et les algorithmes pour minimiser la complexité algorithmique et donc maximiser la performance

Cet EC mobilise de plus les compétences suivantes :

- A5. Traiter des données
- C3. Collecter, stocker et organiser des données biologiques obtenues in vivo, in vitro et in silico y compris massive [big data]

Les connaissances associées à cet EC sont :

- Philosophie de la programmation impérative
- Syntaxe et sémantique
- Notions de variable
- Structures de contrôle
- Notion de fonction
- Notions de base de programmation orientée-objet [encapsulation, classe, méthode, attribut, constructeur]
- Listes, dictionnaires
- Fichiers
- Modularité et réutilisation de code
- Notion de complexité algorithmique
- Utilisation de git

PROGRAMME

Capacités :

- Prédire l'évolution de l'empreinte mémoire d'un algorithme donné
- Evaluer la complexité d'un algorithme
- Concevoir un algorithme pour résoudre un problème
- Implémenter un algorithme en Python dans le style impératif
- Choisir une représentation des données adaptée au problème
- Choisir judicieusement les structures de données et les algorithmes pour minimiser la complexité algorithmique
- Réutiliser du code existant [modules, librairies]
- Passer un programme du style impératif au style orienté-objet
- Versionner son code avec git

INSA LYON

Campus LyonTech La Doua

20, avenue Albert Einstein - 69621 Villeurbanne cedex - France
Tel. +33 (0)4 72 43 83 83 - Fax +33 (0)4 72 43 85 00
www.insa-lyon.fr

Connaissances :

- Encodage des données
- Limites de la représentation des nombres en mémoire
- Principales structures de données algorithmiques (tableaux statiques, tableaux dynamiques, piles, files, tableaux associatifs, arbres, graphes)
- Implémentation de ces types de données abstraits en Python (tuple, list, collections.deque, dict, set...)
- Algorithmes de parcours d'arbres et de graphes
- Notion de complexité algorithmique
- Notion de modularité
- Notion de version de code et commandes git de base
- Notions de base de programmation orientée-objet (encapsulation, classe, méthode, attribut, constructeur)

BIBLIOGRAPHIE

Peter Wentworth, Jeffrey Elkner,
Allen B. Downey and Chris Meyers. How to Think Like a Computer Scientist: Learning with
Python 3
Documentation.

INSA LYON

Campus LyonTech La Doua

20, avenue Albert Einstein - 69621 Villeurbanne cedex - France

Tel. +33 (0)4 72 43 83 83 - Fax +33 (0)4 72 43 85 00

www.insa-lyon.fr