CAMPUS DES MÉTIERS ET DES QUALIFICATIONS
Informatique et électronique de demain - Auvergne-Rhône-Alpes

Séance Codage Vert - Impact environnemental des algorithmes : optimisation et adaptation pour un numérique responsable

Présentation générale

Titre de la ressource : TP Codage vert — Impact environnemental des algorithmes
Type de ressource : Travail pratique sur notebook Jupyter
Durée indicative : 2h
Public cible : BTS CIEL et BTS SIO - 1re et 2e année
Format : Travail en autonomie, seul ou en binôme, avec accompagnement possible de l'enseignant
Niveau attendu : accessible sans connaissances avancées en Python

---

But de la séance

Cette séance s'inscrit dans une démarche de sensibilisation au codage vert. Elle propose d'aborder l'impact environnemental du code informatique à travers un TP en Python, réalisé dans un notebook Jupyter.

L'objectif est de faire comprendre aux étudiants que les choix de programmation ne sont pas neutres. Un algorithme, une recherche dans une base de données ou un programme exécuté trop souvent peuvent influencer le nombre d'opérations réalisées, le temps d'exécution, la consommation d'énergie et donc l'impact environnemental d'un système informatique.

Les étudiants comparent plusieurs solutions techniques, observent les différences de nombre d'opérations, mesurent approximativement des émissions avec CodeCarbon, puis réfléchissent à des pistes d'optimisation et d'adaptation aux usages réels.

Opération soutenue par l'État dans le cadre de l'AMI « Compétences et Métiers d'Avenir » du Programme France 2030, opéré par la Caisse des Dépôts » (La Banque des Territoires).

Objectifs pédagogiques

À l'issue de la séance, les étudiants seront capables de :

1. **Comprendre l'impact environnemental des algorithmes à grande échelle.** Les étudiants prennent conscience qu'un algorithme peut avoir un impact environnemental lorsqu'il est utilisé sur un grand volume de données ou répété de nombreuses fois.
2. **Faire le lien entre complexité, temps d'exécution et consommation d'énergie.** Ils comprennent que le nombre d'opérations réalisées par un programme influence son temps d'exécution et peut donc avoir un effet sur la consommation énergétique.
3. **Adapter les programmes aux usages réels pour limiter les traitements inutiles.** Les étudiants découvrent qu'un programme peut être optimisé non seulement dans son code, mais aussi dans sa fréquence d'exécution et son adaptation aux différents cas d'usage.

Contenu de la ressource

La ressource comprend :

- **un notebook Jupyter contenant le TP** au format `.ipynb`
- **un notebook corrigé** permettant à l'enseignant d'accompagner ou de corriger la séance
- **une fiche explicative** pour comprendre, ouvrir et utiliser un notebook Jupyter

La majorité des manipulations consiste à **exécuter du code existant, observer les résultats, comparer des situations et répondre à des questions**. Peu d'écriture de code est demandée, ce qui rend le TP accessible même si les étudiants ne maîtrisent pas encore parfaitement Python.

Opération soutenue par l'État dans le cadre de l'AMI « Compétences et Métiers d'Avenir » du Programme France 2030, opéré par la Caisse des Dépôts » (La Banque des Territoires).

Déroulement de la séance

Phase 1 — Comparer deux algorithmes de recherche

Les étudiants découvrent un contexte concret : une application de messagerie doit vérifier si un utilisateur est autorisé à se connecter.

Deux méthodes de recherche sont comparées :

- **la recherche linéaire**, qui parcourt les éléments un par un
- **la recherche dichotomique**, qui coupe la zone de recherche en deux à chaque étape, à condition que la liste soit triée

---

Phase 2 — Comprendre la complexité

Les étudiants découvrent ensuite la notion de **complexité temporelle**.

Ils comprennent que ce n'est pas seulement le nombre de lignes de code qui compte, mais surtout le nombre d'instructions réellement exécutées lorsque le programme fonctionne.

Phase 3 — Relier opérations, énergie et impact environnemental

Cette phase permet de comprendre que l'impact environnemental d'un programme dépend aussi de la durée d'exécution, de la puissance utilisée et de la manière dont l'électricité est produite.

Phase 4 — Passage à l'échelle

Les étudiants comparent le nombre théorique d'opérations réalisées par deux algorithmes lorsque la taille des données devient très grande.

Le TP met en évidence que :

- pour un petit **n**, la différence entre deux algorithmes peut sembler faible
- pour un très grand **n**, l'écart entre les algorithmes augmente fortement

Opération soutenue par l'État dans le cadre de l'AMI « Compétences et Métiers d'Avenir » du Programme France 2030, opéré par la Caisse des Dépôts » (La Banque des Territoires).

Phase 5 — Mesurer approximativement avec CodeCarbon

Les étudiants utilisent ensuite **CodeCarbon**, une bibliothèque open source permettant d'estimer les émissions de CO_2 liées à l'exécution d'un code Python.

Ils comparent les émissions estimées pour deux algorithmes de recherche.

L'objectif n'est pas d'obtenir une mesure parfaitement exacte. Les programmes testés sont courts, les valeurs peuvent varier d'un essai à l'autre et les émissions mesurées peuvent être très faibles.

CodeCarbon est donc utilisé comme un **outil de sensibilisation** permettant de comparer deux situations et de faire émerger une tendance : un algorithme moins efficace peut mobiliser davantage de ressources, surtout à grande échelle.

Phase 6 — Optimiser une recherche dans une base de données

Le TP montre ensuite qu'une application ne cherche pas toujours une information dans une simple liste Python. Dans la réalité, elle utilise souvent une **base de données**.

Les étudiants créent une base de données fictive contenant un grand nombre d'utilisateurs, puis comparent deux situations :

- une recherche **sans index**
- une recherche **avec index**

Opération soutenue par l'État dans le cadre de l'AMI « Compétences et Métiers d'Avenir » du Programme France 2030, opéré par la Caisse des Dépôts » (La Banque des Territoires).

Phase 7 — Adapter la fréquence d'exécution aux usages

Dans cette dernière partie, les étudiants découvrent qu'un programme peut aussi consommer inutilement s'il est exécuté trop souvent.

Le TP prend l'exemple d'un serveur qui vérifie les connexions des utilisateurs toutes les minutes.

Les étudiants comparent cette stratégie fixe avec une stratégie adaptée au niveau d'activité.

Bibliographie et ressources

- CodeCarbon — outil open source d'estimation des émissions liées à l'exécution de code. https://codecarbon.io/
- Document pédagogique fourni par Lydie du Bousquet sur le green coding.

Synthèse

Cette séance permet de sensibiliser les étudiants à l'impact environnemental des algorithmes à partir d'exemples concrets de programmation.

Elle montre que plusieurs leviers peuvent contribuer à réduire l'impact d'un programme :

- choisir un algorithme adapté ;
- réduire le nombre d'opérations inutiles ;
- optimiser les recherches dans les bases de données ;
- éviter les exécutions trop fréquentes ;
- adapter les traitements aux usages réels.

À grande échelle, ces choix peuvent réduire la charge des serveurs, la consommation d'énergie et l'impact environnemental d'un service numérique.

Opération soutenue par l'État dans le cadre de l'AMI « Compétences et Métiers d'Avenir » du Programme France 2030, opéré par la Caisse des Dépôts » (La Banque des Territoires).