**Conditions générales pour un bon déroulement du projet**

**A qui s'adresse ce projet ?**

Ce projet s'adresse à des élèves de 1ère NSI au 3ème trimestre.

**Que doivent savoir faire les élèves avant d'aborder ce projet ?**

- Programmer en Python (traiter des données en tables, réaliser des fonctions, importer et utiliser des modules et des bibliothèques);

- Avoir des notions concernant les tris (en quoi ça consiste, avec quelles structures);

- Avoir des notions concernant la complexité temporelle (notion de temps d'exécution dépendant de la taille des données à traiter, notion de classe de complexité ( O(√n), O(log(n)), O(n), O(n.log(n)), O(n²), … ));

- Savoir ce que sont les pré et post-conditions, ce qu'est une assertion et savoir la programmer.

**Combien de temps dure ce projet ?**

3 séances de 2h en classe associées à un peu de travail à la maison.

**Comment les élèves évoluent pour réaliser ce projet?**

- Les élèves de bon niveau pourront avancer à leur rythme en suivant les instructions des documents écrits et quelques informations données par le professeur;

- Les élèves moyens, ou faibles, seront guidés en classes par l'enseignant pour la très grande majorité des travaux à effectuer, et réaliserons avec l'enseignant une exploitation commune des résultats. Des outils pourront être fournis afin que chacun aboutisse à des résultats concrets.

**Objectifs du projet**

**- Comprendre un cahier des charges;**

**- Préparer une ou des fonctions de test en vue de s'assurer du respect d'un cahier des charges d'une future fonction ou procédure à programmer;**

**- Programmer des assertions pour tester des préconditions;**

**- Traduire un pseudo code pouvant être complexe en Python.**

**- Visualiser et "ressentir" la complexité en temps de différents algorithmes;**

**- Voir, par la complexité, l'intérêt d'optimiser un algorithme.**

**Organisation et difficultés à prévenir**

**1ère séance de 2h :**

Présentation de l'ensemble du projet et des objectifs attendus

**Activité fonction test tri** à faire de manière individuelle en classe avec l'aide du professeur.

La fonction à tester sera fourni par le professeur (fonction de tri encapsulant la fonction native sort() de Python).

**Objectifs :**

- Comprendre un cahier des charges;

- Préparer une ou des fonctions de test en vue de s'assurer du respect d'un cahier des charges d'une future fonction ou procédure à programmer;

- Programmer des assertions;

**Difficultés à prévoir :**

- Comprendre le cahier des charges et cerner l'intérêt de faire une fonction test pour l'instant découplée de quelque chose de concret ;

- Penser à réaliser des fonctions nécessaires aux tests des assertions;

- Mettre en œuvre la fonction test réalisée;

**Pas d'évaluation**, la fonction de test réalisée devra être fonctionnelle en fin de séance.

**2ème séance de 2h :**

**Activité fonction tri :** L'activité est à faire par groupe de 3, chacun devant réaliser un des algorithmes de tri avec ou sans l'aide des autres. Les trois fonctions programmées seront pour la prochaine séance regroupées dans un même module.

**Objectifs :**

- Traduire du pseudo code en Python ;

- Déterminer quand il y a besoin de créer des fonctions annexes et les intégrer, si besoin, dans le corps de la fonction de tri;

- Définir et programmer toutes assertions de préconditions;

**Difficultés à prévoir :**

- S'assurer que chacun travaille au moins sa fonction;

- Penser à utiliser la fonction de test précédemment réalisée pour valider la fonction;

- Documenter correctement la fonction réalisée.

Le travail sera à finir à la maison avec une semaine de marge.

Les plus habiles peuvent programmer l'ensemble des fonctions de tri.

**Evaluation :** "Evaluation activite fonction tri.xls"

**3ème séance de 2h :**

**Activité Complexité Temps Tri :** L'activité est à faire par groupe de 3 (les mêmes que pour l'activité fonction tri). Le déroulement de cette activité se fait en étant accompagné par le professeur

**Objectifs :**

- Utiliser une bibliothèque "artisanale";

-"Ressentir " le temps d'exécution en fonction de la taille des données à traiter;

- visualiser par des graphiques l'influence de la taille des données à traiter sur le temps d'exécution, ainsi que l'influence du type d'algorithme sur ce temps ;

- visualiser par des graphiques que le temps d'exécution d'un algorithme est directement lié au nombre d'opérations unitaires effectuées ;

- visualiser par des graphiques ce qu'est une classe de complexité en temps d'un algorithme.

**Difficultés à prévoir :**

- Utiliser la bibliothèque "complexite\_temps";

- Manipuler un tableur pour regrouper les données de différents fichiers csv et tracer des graphiques explicites;

- Comprendre, par la lecture du code de la fonction "mesure\_temps", comment on peut mesurer le temps d'exécution d'une fonction.

- Etablir un protocole pour estimer la classe de complexité de la fonction native sort().

Une mise en commun des données est effectuée, et un fichier peut être distribué contenant les résultats des expérimentations et leur mise en forme.

**Evaluation :** des questions portant sur des connaissances générales sur la notion de complexité seront intégrées au prochain QCM.