ENSEIGNEMENT ANNUEL D'INFORMATIQUE ET ALGORITHMIQUE

I - Programme du premier semestre.

Les séances de travaux pratiques du premier semestre poursuivent les objectifs suivants :

- *consolider l'apprentissage de la programmation qui a été entrepris dans les classes du lycée en langage Python ;*
- *mettre en place une discipline de programmation : découpage modulaire à l'aide de fonctions et programmes, annotations et commentaires, évaluation par tests ;*
- *mettre en pratique des algorithmes facilitant le traitement de l'information, la modélisation, la simulation.*

1 - Algorithmique des listes

Il s'agit de présenter des algorithmes simples, spécifiés de façon abstraite, puis de les traduire dans un langage de programmation, ici Python. On utilisera ces activités pour construire une progression pour assimiler les notions de variables, de type, d'affectation, d'instruction conditionnelle, de boucles conditionnelles ou inconditionnelles et manipuler de façon simple les listes en Python. S'il n'est pas souhaitable de les formaliser, on dégagera de l'étude de ces algorithmes simples les problématiques de la correction et la terminaison des algorithmes. Ces notions ne sont pas exigibles.

Recherche séquentielle dans une liste.	Recherche d'un élément. Recherche du maximum, du second maximum.
Algorithmes opérant sur une structure séquentielle par boucles imbriquées.	Recherche des deux valeurs les plus proches dans une liste.
Algorithmes dichotomiques.	Recherche dichotomique dans une liste triée.
Algorithmes gloutons.	Rendu de monnaie. Allocation de salles pour des cours.

2 - Statistiques descriptives et analyse de données.

Dans ce paragraphe, on analysera des données statistiques publiques obtenues sous forme d'un fichier csv (Comma-separate-value). Pour ce faire, on pourra utiliser le site data.gouv ou le site de l'Insee et choisir des données socio-économiques. On travaille directement sur le fichier de données sous forme de table en important la bibliothèque `pandas` *ou après transformation directement sur un tableur. On indiquera aux étudiants les commandes à utiliser. Aucune de ces commandes de cette bibliothèque n'est exigible.*

Lecture d'un fichier de données simples. Notion de descripteur.	
Exemples d'analyse des données.	On pourra faire des tris sélectifs, donner des exemples de calculs d'indicateurs de position : moyenne, médiane, mode, quantiles. ou d'indicateurs de dispersion : étendue, variance et écart-type empiriques, écart inter-quantile. On discutera la signification des résultats obtenus.
Représentations des données. Diagrammes en bâtons, histogrammes.	On pourra utiliser la bibliothèque `matplotlib`.

3 - Approximation numérique

Calcul approché de la racine d'une équation du type $f(x) = 0$.	On utilisera différentes méthodes dont certaines résulteront d'une étude mathématique (suites récurrentes, encadrements, dichotomie).

II - Programme du deuxième semestre.

1 - Graphes finis, plus courts chemins

Il s'agit de revenir sur le modèle des graphes et d'étudier les démarches algorithmiques permettant de les analyser selon leurs représentations.

Graphes.	Un graphe est implémenté à l'aide de listes d'adjacence (rassemblées par exemple dans une liste ou dans un dictionnaire) ou par sa matrice d'adjacence.
Recherche d'un plus court chemin dans un graphe pondéré avec des poids positifs. Algorithme de Dijkstra.	

2 - Simulation de phénomènes aléatoires

Simulation d'expériences aléatoires élémentaires conduisant à une loi usuelle. Simulation de phénomènes aléatoires.	Loi binomiale, loi géométrique.

III - Annexe : Langage Python

Toute la richesse du langage Python ne peut pas être entièrement maîtrisée par un étudiant, aussi le paragraphe ci-dessous liste limitativement les éléments du langage Python (version 3 ou supérieure) dont la connaissance est exigible des étudiants. Il s'agit de la liste des commandes utiles pour les travaux pratiques des deux années de formation. Il n'y a pas lieu d'introduire en première année les commandes qui relèvent de notions de seconde année.

1 - Types de base

`+` `-` `*` `/` `**`	Opérations arithmétiques de base.
`==` `>` `<` `>=` `<=` `!=`	Comparaison, test.
`True` `False` `and` `or` `not`	Logique.
`from ... import *, import ... as`	Importation d'une bibliothèque.

2 - Structures de contrôle

Instruction d'affectation =.

Instruction conditionnelle `if`, `elif`, `else`.

Boucle `for` ; Boucle `while`.

Définition d'une fonction `def` $f(p_1, \ldots, p_n)$ `return`.

3 - Listes

Tableau unidimensionnel ou liste. Définitions d'une liste avec une boucle ou en compréhension. Fonction `range`. Commandes `append` , `len`.	Il n'y a pas lieu de distinguer ces deux structures de données en langage Python.
Recherche séquentielle dans une liste. Commandes `in del count`.	
Manipulations élémentaires de listes. Commandes + et ∗.	

4 - Utilisation de modules, de bibliothèques

`from ... import *, import ... as`	Importation d'une bibliothèque.

Pour le calcul numérique, le traitement statistique ou la simulation de phénomènes aléatoires, certaines bibliothèques s'avèrent utiles. Elles sont listées ci-dessous avec les fonctions pertinentes. Toute utilisation d'une telle fonction doit obligatoirement être accompagnée de la documentation utile, sans que puisse être attendue une quelconque maîtrise par les étudiants de ces éléments.

a) Dans la bibliothèque `numpy`

Exemple d'importation : `import numpy as np`

`np.array, np.zeros, np.ones, np.eye, np.linspace, np.arange`	Création de vecteurs et de matrices. Extraction ou modification d'un élément, d'une ligne ou d'une colonne d'une matrice.
`+` `-` `*` `/` `**`	Opérations arithmétiques de base : coefficient par coefficient.
`==` `>` `<` `>=` `<=` `!=`	Comparaison de deux matrices (`M == N`), comparaison d'une matrice et d'un nombre (`M>=1`).
`a,b = np.shape(M)`	Taille de la matrice `M`.
`np.dot, np.transpose`	Syntaxes exigibles : `np.transpose(M)`, `np.dot(M1,M2)`. L'usage de méthode comme `M.transpose(), M1.dot(M2)` est non-exigible.
`np.sum, np.min, np.max, np.mean, np.cumsum, np.median, np.var, np.std`	Ces opérations peuvent s'appliquer sur une matrice entière ou bien pour chaque colonne (ou chaque ligne). Exemple : `mean(M), mean(M,0), mean(M,1)`
`np.exp, np.log, np.sqrt, np.abs, np.floor`	Ces fonctions peuvent s'appliquer à des variables numériques ou vectoriellement (à des matrices ou vecteurs) élément par élément. On pourra utiliser la commande `f = np.vectorize(f)` mais elle n'est pas exigible.

`np.e, np.pi`

b) Dans la librairie `numpy.linalg`

Exemple d'importation : `import numpy.linagl as al`

`al.inv, al.rank, al.matrix_power,`
`al.solve, al.eig`

c) Dans la librairie `numpy.random`

Exemple d'importation : `import numpy.random as rd`

`rd.random, rd.binomial, rd.randint,` `rd.geometric, rd.poisson,` `rd.exponential, rd.normal, rd.gamma`	On utilisera ces fonctions pour générer un nombre aléatoire ou bien un vecteur ou une matrice à coefficients aléatoires. Exemple : `rd.binomial(10,0.2)`, `rd.binomial(10,0.2,100)`, `rd.binomial(10,0.2,[100,10])`

d) Dans la librairie `matplotlib.pyplot`

Exemple d'importation : `import matplotlib.pyplot as plt`

`plt.plot, plt.show`	Représentations graphiques de fonctions, de suites. On pourra utiliser les commandes `xlim, ylim, axis, grid, legend` mais elles ne sont pas exigibles.
`plt.hist, plt.bar, plt.boxplot`	Représentations statistiques.
Utilisation de la fonction `rd.random` pour simuler des expériences aléatoires.	On pourra simuler ainsi des lois binomiale et géométrique.
Simulation d'échantillons de lois usuelles.	On pourra utiliser les fonctions `rd.binomial, rd.randint, rd.geometric, rd.poisson`

e) Dans la librairie `pandas`

Exemple d'importation : `import pandas as pd`

`pd.read_csv, head, shape, pd.describe`	Pour créer une table à partir du fichier de données et en visualiser ou manipuler une partie.
`pd.mean, pd.std, pd.median, pd.count,` `pd.sort_values.`	Indicateurs statistiques.

ENSEIGNEMENT ANNUEL D'INFORMATIQUE ET ALGORITHMIQUE

I - **Programme du troisième semestre.**

1 - Bases de données

L'administration, les banques, les assurances, les secteurs de la finance utilisent des bases de données, systèmes d'informations qui stockent dans des fichiers les données nombreuses qui leur sont nécessaires. Une base de données relationnelle permet d'organiser, de stocker, de mettre à jour et d'interroger des données structurées volumineuses utilisées simultanément par différents programmes ou différents utilisateurs. Un logiciel, le système de gestion de bases de données (SGBD), est utilisé pour la gestion (lecture, écriture, cohérence, actualisation...) des fichiers dans lesquels sont stockées les données. L'accès aux données d'une base de données relationnelle s'effectue en utilisant un langage informatique qui permet de sélectionner des données spécifiées par des formules de logique, appelées requêtes d'interrogation et de mise à jour.
L'objectif est de présenter une description applicative des bases de données en langage de requêtes SQL (Structured Query Language). Il s'agit de permettre d'interroger une base présentant des données à travers plusieurs relations. On introduira les concepts à l'aide d'exemples simples issus de contextes appropriés (fichier clients, gestion des stocks, gestion du personnel ...)

Modèle relationnel : relation, attribut, domaine, clef primaire « PRIMARY KEY », clef étrangère « FOREIGN KEY », schéma relationnel.	
Vocabulaire des bases de données : table, champ, colonne, schéma de tables, enregistrements ou lignes, types de données.	On s'en tient à une notion sommaire de domaine : entier « INTEGER », chaîne « TEXT ».
Lecture d'un fichier de données simples. Notion de descripteur.	Lecture d'un fichier de données simples. Notion de descripteur.
Opérateurs arithmétiques $+$, $-$, $*$.	
Opérateurs de comparaison : $=$, $<>$, $<$, $<=$, $>$, $>=$.	
Opérateurs logiques : « AND », « OR », « NO ».	

a) Commandes exigibles

« WHERE »	
« SELECT nom_de_champ FROM nom_de_table ».	Sélection de données dans une table.
« INSERT INTO nom_de_table ».	Insertion de données dans une table. On pourra utiliser « VALUES (élément1, élément2,...)".
« DELETE FROM nom_de_table ».	Suppression de données d'une table.
« UPDATE nom_de_ table ».	Mise à jour de données d'une table.

« SELECT* FROM nom_de_table_1 INNER JOIN nom_de_table_2 ».	Réalisation d'une jointure. On pourra ajouter une condition « ON Φ »dans le cas où Φ est une conjonction d'égalités. Aucune autre notion de jointure n'est dans ce programme.
« CREATE TABLE nom_de_table ».	Création d'une table.

b) Commandes non exigibles

On pourra utiliser par commodité la liste d'opérateurs, fonctions et commandes ci-dessous. Ce ne sont pas des attendus du programme et ils sont non exigibles.

Les opérateurs ensemblistes : union « UNION », intersection « INTERSECTION », différence « EXCEPT ».

Les opérateurs spécifiques de l'algèbre relationnelle : projection, sélection (ou restriction), renommage, produit cartésien .

Les fonctions d'agrégation : min « MIN », max « MAX », somme « SUM », moyenne « AVG », comptage « COUNT ».

Les commandes « DISTINCT », « ORDER BY »

2 - Equations et systèmes différentiels

L'objectif est d'illustrer les concepts vus dans le cours de mathématiques. On pourra dégager sur des exemples simples des notions qualitatives, mais aucune technicité n'est attendue. La discrétisation d'une équation différentielle n'est pas au programme. On pourra utiliser le solveur scipy.integrate.odeint ; la maîtrise d'un tel outil n'est pas exigible.

Représentations graphiques de trajectoires.	Sur des exemples en lien avec le programme : Interprétation des paramètres. Influence des conditions initiales. On observera le phénomène de convergence vers un équilibre.

3 - Statistiques descriptives bivariées

Série statistique à deux variables, nuage de points associé. Point moyen $(\bar{x}, \bar{y})$ du nuage. Covariance empirique, cœfficient de corrélation empirique, droites de régression.	On tracera le nuage de points et les droites. de régression et on pourra effectuer des prétransformations pour se ramener au cas linéaire. On distinguera les variables explicatives des variables à expliquer.

II - **Programme du quatrième semestre.**

1 - Chaînes de Markov

Ce thème sera l'occasion de revoir les simulations de lois discrètes, de revisiter les notions de programmation et de représentation de données par un graphe fini, qui sont vues en première année, ainsi que d'appliquer les résultats et techniques d'algèbre linéaire étudiés au troisième semestre.

Matrice de transition. Étude sur des exemples simples. Etat stable. Comportement limite.	On pourra étudier par exemple l'indice de popularité d'une page Web (PageRank), modéliser l'évolution d'une société (passage d'individus d'une classe sociale à une autre), ou les systèmes de bonus-malus en assurances. Simulation et mise en évidence d'états stables. On observera la convergence en loi d'une chaîne de Markov (sur un graphe complet) vers son état stable.

2 - Estimation ponctuelle ou par intervalle de confiance

Comparaison de différents estimateurs ponctuels d'un paramètre.	On pourra utiliser des données issues de situations réelles (simple comparaison de valeurs numériques) ou créer plusieurs jeux de données par simulation. Dans ce dernier cas, on pourra comparer les lois des estimateurs par exemple à l'aide d'histogrammes.
Intervalle de confiance asymptotique obtenu avec le théorème limite central pour estimer le paramètre d'une loi de Bernoulli.	Résultat admis On pourra comparer, en majorant $p(1-p)$ par $\frac{1}{4}$, les intervalles de confiance obtenus par l'inégalité de Bienaymé-Tchebychev, et les intervalles de confiance asymptotiques obtenus par l'approximation de la loi binomiale par la loi normale. La comparaison pourra se faire en calculant les demi-largeurs moyennes des intervalles et leurs niveaux de confiance.