Plan et supports de l'atelier. L'activité A sera menée en commun, les autres activités pourront être menées en parallèle selon le choix des participants.

A. La notion de fonction en seconde générale. (Inspiré des nombreux exercices d'algorithmiques figurant dans le manuel Pixel de chez Bordas, auxquels j'ajoute toujours une activité graphique).

Les algorithmes sont élaborés et mise en œuvre d'abord en lignes de commandes, et une fois au point les lignes de commandes sont intégrées dans une ou plusieurs fonctions. On compare les différents aspects des fonctions en programmation, avec les fonctions mathématiques. La représentation graphique confronte les élèves à la nécessité de raisonner sur les points et leurs coordonnées.

A.1. Fonctions affines (FonctionAffinePosi2nde0.py)

Les coefficients sont les arguments de la fonction, qui est définie dans le corps du programme.

A.1.a. Représentation graphique du signe d'une fonction affine **f(x)** = **mx + p** sur **[a ; b]**. (SigneAffine3(...) dans FonctionAffinePosi2nde0.py)

L'objectif est de différencier des points d'ordonnées négatives de ceux d'ordonnées positives, soit par des couleurs, soit par des marques. Il faut d'abord créer les listes des valeurs des abscisses et des ordonnées, ce qui montre la nécessité de bien décrire le domaine de définition. On crée ensuite une liste qui contient les signes des ordonnées et enfin une liste contenant les couleurs (par exemple bleue pour les ordonnées négatives, verte pour les ordonnées positives) à appliquer aux points selon le signe de leur ordonnée.

A.1.b. Représentation graphique de la position de la droite représentative de la fonction affine **f** sur **[a ; b]** par rapport à la droite d'équation **y = k**, (toujours décimal en informatique). (PosikAffine4(...) dans FonctionAffinePosi2nde0.py)

L'objectif est de représenter différemment les points d'ordonnées inférieures à **k** de ceux d'ordonnées supérieures à **k**, soit par des couleurs, soit par des marques. C'est une simple petite modification par rapport au programme du **A.1.a.** On créera un nouveau fichier.

A.1.c. Soit **f** et **g** deux fonctions affines définies sur l'intervalle **[a ; b]**. (PosikAffine5(...) dans FonctionAffinePosi2nde0.py)

L'objectif est d'illustrer les positions relatives des deux droites **Df** et **Dg** représentatives des fonctions affine **f** et **g**.

Il s'agit donc de représenter différemment des points de **Dg** d'ordonnées inférieures à ceux de **Df** de même abscisse, de points de **Dg** d'ordonnées supérieures à ceux de **Df** de même abscisse, soit par des couleurs, soit par des marques. On peut facilement modifier le programme du **A.1.b.** pour obtenir les résultats demandés.

A.2. Fonctions quelconques (FonctionGenePosi2nde0.py)

A.2.a. et **A.2.b.** Reprendre, respectivement, les exercices **1.b.**, **1.c.**, en utilisant les fonctions quelconques fournies dans les énoncés. (IneqFgene2(...), IneqFgene4(...) dans FonctionGenePosi2nde0.py)

Il est plus simple, dans un premier temps, de définir les fonctions en dehors du programme principal. Les adaptations des programmes du **A.1**. sont simples à faire.

B. Un intervalle de fluctuation d'une variable binomiale – Première S (BinoDistribEtIF.py)

B.1. Détermination de l'intervalle de fluctuation d'après l'algorithme du document ressource de première S. (IfBino(…) dans BinoDistribEtIF.py)

Il faut construire des listes contenant les valeurs de la distribution et de la répartition de la variable aléatoire binomiale. On cherche, dans la liste répartition, les valeurs seuil 0,025 et 0,975 (IF au seuil de 0,95) et on repère les indices correspondant qui seront les valeurs recherchées, a et b, de la variable aléatoire binomiale. On programme une présentation propre des résultats.

B.2. Illustration graphique de cet intervalle de fluctuation. (IfBino(…) dans BinoDistribEtIF.py)

L'objectif est de représenter la répartition de la v.a. binomiale par un diagramme en barres avec des barres de couleurs différentes pour les valeurs de la v.a. appartenant à l'IF. On utilisera le même principe que celui vu au **A.**

B.3. Estimation d'un intervalle de fluctuation d'une variable aléatoire par simulation

Lorsque l'on n'a pas pu modéliser la distribution de la variable aléatoire par une loi binomiale ou une autre loi, la simulation est une bonne technique pour calculer un intervalle de fluctuation.

C. Problème historique d'un duc de Toscane – première S terminale S (JeuDeDe0.py)

Nous allons prolonger le problème historique en recherchant la distribution de la variable aléatoire S somme des valeurs obtenues en jettant aléatoirement 3 dés équilibrés. À partir du résutat de cette distribution nous allons pouvoir répondre au paradoxe soulevé par le problème historique. Nous verrons deux façons de travailler sur cette distribution : par la simulation et par l'obtention de la distribution "exacte".

C.1. Une simulation

L'algorithme enregistre les nbsim lancers et le résultat de la somme S dans une liste qui est ensuite exploitée pour en tirer le tableau des effectifs simulés de S.

C.2. La distribution "exacte"

L'algorithme détermine l'ensemble de tous les triplets possibles, en créant une liste par compréhension sur trois indices. Cette liste est ensuite traitée pour en tirer le tableau de distribution de la variable aléatoire S.

D. Marches aléatoires – seconde G, première S terminale S

D.1. Le loup et l'agneau (LA0(…) dans LoupAgneau0.py)

Cette activité est inspirée de deux articles parus dans deux bulletins de l'APMEP : Bulletin de l'APMEP. Num. 515. p. 401-406 – Le loup et l'agneau – Une activité de probabilités en seconde et Bulletin de l'APMEP. Num. 516. p. 637-639 – Le loup et l'agneau (cas général).

Un Agneau et un loup se trouvent respectivement aux points O(0 ; 0) et N(n ; n) d'un carré de diagonale ON dans un repère orthonormé. Ils jettent à tour de rôle, au hasard, n fois, une pièce équilibrée. Si l'agneau obtient **pile**, il se déplace de **1 en x sinon de 1 en ordonnée**. Si le loup obtient **pile**, il se déplace de **–1 en x sinon de –1 en ordonnée**.

Nous verrons comment :

- Simuler un jeu (n déplacements) et en faire la représentation graphique.
- Simuler nbsim fois ce jeu et estimer la probabilité de rencontre.
- Estimer la distribution de probabilité des points de rencontre sur un carré n × n.

D.2. La marche sur le pont avec le robot TOM – Terminale S (TrompetteRoboTom.odt)

Cette activité est inspirée de l'**exercice 4 du sujet 2013 de septembre d'Antilles-Guyane TS**. Un robot se déplace de façon aléatoire sur un pont sans garde-corps. L'exercice propose un algorithme simulant la position du robot au bout de x déplacements.

Après avoir analysé et programmé en Python, l'algorithme proposé, nous verrons comment simuler et représenter graphiquement des marches aléatoires du robot puis estimer et calculer la probabilité que le robot passe le pont sans tomber à l'eau.

E. Exemples d'intégration numérique – Sommes de Darboux – Terminale S (PolynésieAlgoBac2013.odt)

Cette activité est inspirée de l'**exercice 1-2. du sujet de juin 2013 de Polynésie TS**. L'exercice propose un algorithme permettant de calculer une valeur approchée de l'aire d'un domaine sous la courbe d'une fonction continue et strictement décroissante, par la méthode des grands rectangles.

Après avoir analysé et programmé en Python, l'algorithme proposé, nous verrons comment l'agrémenter d'une représentation graphique puis le prolonger en considérant le cas d'une fonction strictement croissante, le cas d'une fonction continue quelconque, le cas de la méthode des petits rectangles et conclure par un petit rappel sur les sommes de Darboux.