Répétition d'expériences de Bernoulli

Enoncé

Une entreprise fabrique des petites voitures et les vend par lot de 100. La probabilité qu'un jouet produit présente un léger défaut de peinture a été établi à 28%.

L'entreprise examine un lot et trouve 35 % de jouets présentant un défaut. Elle se demande si sa chaîne de production est dégradée.

```
Fonction JouetDefaut(p,t)
  compteur ← 0
  Pour i allant de 0 à t-1
      Si Nombre Aleatoire < p alors
            compteur ← compteur + 1
      Fin Si
      Fin Pour
      Renvoyer compteur/t
```

1. Pourquoi la situation présentée peut se modéliser par une répétition d'expériences de Bernoulli ?
2. Réaliser la simulation en Python de la situation. Pour cela retranscrire la fonction **JouetDefaut** de paramètres **p** et **t** où **p** représente la probabilité de présenter un défaut et où **t** représente la taille du lot. La fonction renvoie la fréquence de pièces défectueuses.
3. On souhaite répéter plusieurs fois la situation précédente en stockant l'ensemble des fréquences obtenues pour pouvoir les étudier et prendre une décision. Définir en Python la fonction **Echantillon** de paramètre **p, t** et **n** où **p** représente la probabilité de présenter un défaut, **t** représente la taille du lot et **n** représente le nombre de lots testés. La fonction renverra la liste des fréquences obtenues. On pourra représenter le nuage de points des fréquences obtenues.

1. Répétition d'expériences de Bernoulli

La situation consiste à examiner chacune des pièces d'un lot.

Pour chaque pièce, deux issues possibles : défectueuse ou pas défectueuse, avec une probabilité déclarée de 28%. On a une épreuve de Bernoulli.

La répétition consiste à examiner chacune des petites voitures du lot pour déterminer si elle est ou non défectueuse.

On a donc bien la répétition d'expériences de Bernoulli.

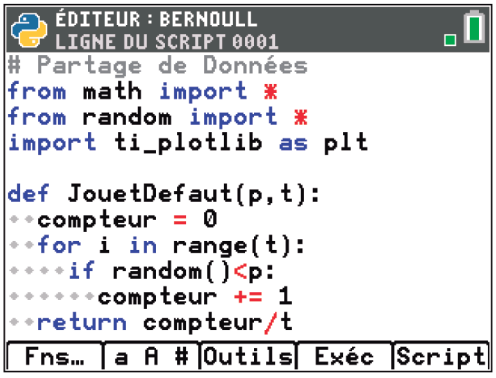
2. Définition de `JouetDefaut` en Python

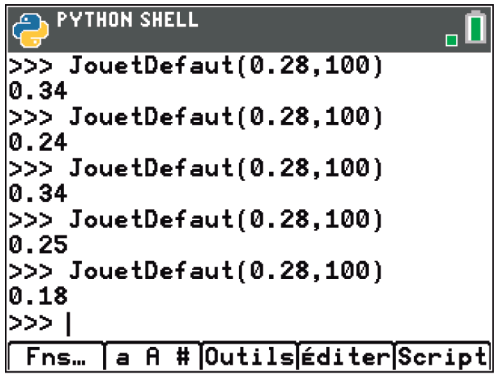
On crée un script **BERNOULL** dans lequel on importe la librairie **math**, la librairie **random** et la librairie **ti_plotlib** dont on raccourcit le nom en **plt**. Il s'agit d'une librairie de représentation graphique dont on se servira en fin d'exercice.

Dans le code de notre fonction **JouetDefaut**, nous réutilisons des principes déjà rencontrés dans les fiches précédentes, à savoir la notion de compteur pour cumuler les réussites (ici des pièces défectueuses). Ce nombre est divisé à la fin par le nombre d'expériences réalisées pour retourner la fréquence des succès. L'instruction **random()<p** permet de simuler la probabilité passée en paramètre et ce, **t** fois pour la taille du lot. On teste notre fonction, en console (Exéc), à l'aide de la commande **JouetDefaut(0.28,100)**.

On constate la variation des fréquences d'un lot à l'autre.

On souhaite maintenant conserver une trace des différentes fréquences obtenues.





Répétition d'expériences de Bernoulli

3. Définition de **Echantillon** en Python

On complète notre script avec la fonction **Echantillon**. Elle consiste à stocker dans une liste les résultats renvoyés par notre fonction **JouetDefaut** précédente, avec les paramètres **p** et **t**, ceci **n** fois, le paramètre **n** correspondant au nombre de lots que nous souhaitons prélever.

Là encore, nos travaux précédents nous ont conduit à voir plusieurs possibilités de codage de cette situation. C'est pour cela que nous vous présentons deux solutions. La première consiste à produire un code très court dans lequel nous définissons la liste des fréquences par « compréhension » à l'aide de l'instruction **[JouetDefaut(p,t) for i in range(n)]**. Chaque élément est le résultat de la fonction **JouetDefaut** appelée **n** fois.

L'autre solution, plus classique, consiste à construire au fur et à mesure la liste par ajout successif du résultat de la fonction **JouetDefaut**. La boucle n'est plus à l'intérieur de la définition de la liste mais dans une structure, plus habituelle, à l'extérieur.

Pour la suite, nous modifions **JouetDefaut** pour qu'elle renvoie la fréquence en pourcentage, arrondi à l'unité, à l'aide de l'instruction **return int(compteur/t*100).** Elle s'appelera **JouetDefaut1.**

Nous créons la fonction **Echantillon2**, à partir de la fonction **Echantillon**. La liste **l** s'appelle désormais **ly** et nous créons une liste des **n** premiers entiers stockés dans **lx** à l'aide de l'instruction **[i for i in range(n)]**.

Nous pouvons maintenant préparer l'environnement graphique pour représenter le nuage de points (x;y) construit à partir des éléments respectifs de **lx** et **ly**. Vous trouverez dans le module **ti_plotlib** toutes les instructions nécessaires pour calibrer automatiquement la fenêtre graphique en prévision des points à représenter, afficher les axes de la représentation, dessiner en rouge la droite d'équation y = 35, dessiner en vert la droite d'équation y = 28 et enfin le nuage de points, en bleu, à l'aide de l'instruction **plt.scatter(lx,ly,"")**.

On affiche enfin le graphique via l'instruction **plt.showplot**. Lors de l'exécution, il faut appuyer ensuite sur la touche annul pour quitter la représentation graphique et afficher la liste des fréquences en pourcentage.

On parvient à se convaincre que 35% de jouets défectueux, bien que dans une limite haute, peut se produire plusieurs fois.