**TD N° 13 : révisions**

1. **Suites de Syracuse**

Une suite de Syracuse est définie par :



1. Écrire une fonction récursive permettant de calculer le nième terme de la suite de Syracuse dont le premier terme vaut 7. Étendre cette fonction pour qu’elle admette un premier terme quelconque en paramètre.
2. Ecrire une fonction itérative permettant de réaliser le même calcul.
3. Écrire une fonction qui retourne le rang du premier terme valant 1 (s’il existe). On se limitera aux 100 premiers termes.
4. **Paradoxe des anniversaires**

Lorsqu’on réunit 23 personnes, il y a une chance sur 2 pour que 2 d’entre elles soient nées le même jour. On souhaite vérifier expérimentalement ce résultat contre-intuitif, appelé « paradoxe des anniversaires ». Une date anniversaire sera un nombre tiré aléatoirement entre 1 et 365.

1. Écrire une fonction booléenne doublon(t) qui retourne True si le tableau t contient un doublon et False sinon.

On pourra utiliser pour cela un ensemble stockant les différentes valeurs rencontrées.

1. Pour vérifier expérimentalement le paradoxe des anniversaires, on se propose de répéter un grand nombre de fois (par exemple 1000) le tirage aléatoire de 23 dates.

Environ la moitié des tirages (500) devrait alors comporter au moins un doublon.

Écrire le programme correspondant.

1. **Comptes bancaires**
2. définissez une classe CompteBancaire() avec deux attributs (nom et solde) et trois méthodes :
* depot(somme) : ajout de somme au solde
* retrait(somme) : retrait de somme au solde
* affiche() : affiche le nom du titulaire et le solde du compte
1. Écrire un exemple d’utilisation comportant par exemple les étapes suivantes :
* création du compte de Dupont avec un solde de 800 euros
* dépôt de 500 euros sur ce compte
* affichage du solde
1. Créer une classe Banque constituée d’un tableau de comptes, de la méthode ajouteCompte(compte), et de la méthode affiche() qui affiche tous les comptes d’une banque.
2. On souhaite maintenant pouvoir sauvegarder une banque dans un fichier (banque.txt par exemple). Chaque compte correspondra à une ligne du fichier, constituée du nom, suivi d’un séparateur (e. g. point-virgule) et du solde.

Ajouter une méthode toString() à la classe CompteBancaire pour pouvoir constituer cette ligne.

Ajouter la méthode sauve à la classe Banque.

1. Écrire la fonction createBanque permettant de créer une banque à partir d’un fichier

Exemple d’utilisation :

 c1 = CompteBancaire('Dupont', 800)

    c1.depot(500)

    c1.affiche()

    c1.retrait(500)

    c1.affiche()

    print(c1.toString())

    filename = 'banque.txt'

    banque = createBanque(filename)

    banque.affiche()

    banque.ajouteCompte(c1)

    banque.affiche()

    banque.sauve('banque2.txt')

1. **Triangle de Pascal**

Le triangle de Pascal (nommé ainsi en l'honneur de Blaise Pascal) permet de représenter les coefficients binomiaux dans un triangle. Par exemple, pour n=5, le triangle est le suivant :

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

On remarquera que les première et dernière valeurs de chaque ligne sont égales à 1. Chaque autre valeur est égale à la somme des deux valeurs adjacentes dans la ligne précédente.

On souhaite construire ce triangle à l'aide d'une fonction récursive triangle\_pas prenant deux entiers, ligne et colonne, en entrée et renvoyant la valeur du triangle de Pascal pour la ligne et la colonne données.

*Exemples :*

triangle\_pas(1,1)=1 triangle\_pas(3,2)=3 triangle\_pas(5,3)=10

1. Écrire la fonction triangle\_pas
2. En déduire un programme permettant d'afficher à l'écran le triangle de Pascal jusqu'à la ligne *n*, *n* étant entré au clavier par l'utilisateur.