Liste d’exercices

1. **Itératif**

1. Écrire une fonction waitInteger qui renvoie vrai si l'entrée de l’utilisateur est un entier, faux sinon.

print(waitInteger()) → true

def waitInteger():

        try:

            x = int(float(input("Entrer un nombre : ")))

            return True

        except ValueError:

            return False

1. Écrire une fonction isBissextile qui calcule si une année est bisextile ou non.

Une année est bissextile si l'année est divisible par 4 et non divisible par 100 ;

si l'année est divisible par 400 (« divisible » signifie que la division donne un nombre entier, sans reste).

print(isBissextile(2000)) → true

def isBissextile(annee):

    return (annee%4==0  and annee%100!=0) or annee%400==0

1. Écrire une fonction isValidCardNumber qui calcul si un numéro de carte bancaire est valide ou non à l’aide de l’algorithme de luhn.

print(isValidCardNumber(8532)) → true

L'algorithme de Luhn commence par la fin du numéro. Prendre le dernier chiffre et se déplacer vers la gauche. Doubler la valeur de tous les chiffres de rang pair. Si le double d'un chiffre est supérieur ou égal à 10, le remplacer par la somme de ses chiffres. Réaliser la somme

de tous les chiffres obtenus.

Exemple:

Pour le nombre 8532,

Doubler le 8 (8\*2=16 et 1+6=7) pour obtenir 7.

Prendre le 5, non doublé.

Prendre le 3, le doubler pour obtenir 6.

Prendre le 2, non doublé.

La somme est 6+5+7+2 = 20. Comme 20 modulo 10 = 0 → le nombre est bien valide.

Donc 8532 est valide mais pas 8531, ni 8533 ni 3, etc.

def sommeChiffre(nb):

    return nb%10 + nb//10

def testValidLuhn(nombre):

    liste=[]

    #On transforme le nombre en liste.

    while nombre!=0:

        liste.append(nombre%10)

        nombre=nombre//10

    #Attention 8532 -> [2,3,5,8]

    somme=0

    for i in range(len(liste)):

        if i%2 != 0:

            somme += sommeChiffre(liste[i]\*2)

        else:

            somme += liste[i]

    return somme%10 ==0

1. **Listes**

1. Insérer un élément à la bonne place une liste triée en utilisant la recherche dichotomique.

def rechOrInsertDicho(elem, liste):

    begin=0

    end=len(liste)

    while True:

        mid = (begin+end)//2

        if (elem>liste[mid]):

            begin = mid

            if (mid== (begin+end)//2):

                break

        elif (elem<liste[mid]):

            end = mid

            if (mid==(begin+end)//2):

                break

        else :

            return mid

1. Écrire une fonction qui supprime les éléments en double d'une liste.

Exemple d'exécution :

liste de départ : L=[1,2,5,8,6,2,5,9,1,8,8] → [1, 2, 5, 8, 6, 9]

def deleteRepetition(liste):

    cleanListe = []

    for i in liste:

        if not (i in cleanListe):

            cleanListe.append(i)

    return cleanListe

1. Écrire une fonction flatten qui attend une liste L et qui crée une liste L1 qui contienne les mêmes éléments que L, mais à plat.

Exemple :

si L = [1,2,[3,4],5,6,[7,[8,[9,10,[11,12],13],14,15],16,17,18],19,[20,21,[22,23,24,[25,26,27],28],[29],30],31,32]

Alors, en fin de programme :

L1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32]

def flatten(liste):

    flat=[]

    for i in liste:

        if type(i) == list:

            flat.extend(flatten(i))

        else:

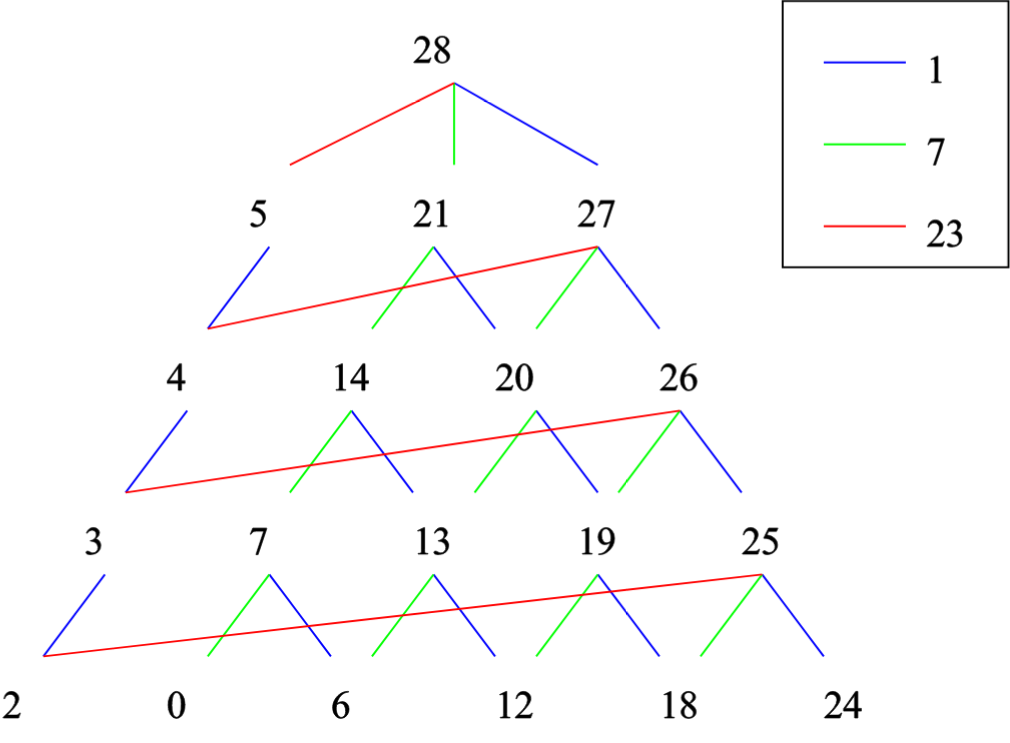
            flat.append(i)

    return flat

1. **Récursif**

1. Algorithme de rendu de monnaie

Trouver la meilleure manière (avec le moins de pièces de rendre la monnaie).



Dans un premier temps, on voudrait savoir combien de pièce sont nécessaire pour rendre la monnaie sur la somme prévu. Rendu([1,7,23],28) → 4 (car il faut minimum 4 pièce pour rendre 28 euros)

Ensuite, on va aussi donner la valeur des pièces que nous avons utilisé pour cela : rendu([1,7,23],28) → [0,4,0] car la meilleure façon de rendre 28 euros avec des pièces de 1, 7 et 23 est de rendre 4 pièces de 7.

https://fr.wikipedia.org/wiki/Probl%C3%A8me\_du\_rendu\_de\_monnaie

def renduMonnaie(liste, somme):

    if somme in liste:

        return 1

    else :

        result=[]

        for i in liste:

            if somme -i >0:

                result.append(renduMonnaie(liste,somme-i))

        return 1 + min(result)

Version avancé :

def indiceMin(liste):

    var=liste[0]

    indiceMin=0

    for i in range(len(liste)):

        if liste[i]<var:

            indiceMin=i

            var=liste[i]

    return indiceMin

def sommePiece(liste):

    var=0

    for i in liste:

        var+=i

    return var

def renduMonnaies(pieces, somme):

    if somme == 0:

        return [0]\*len(pieces)

    else :

        result=[]

        for i in range(len(pieces)):

            if somme - pieces[i] >= 0:

                result.append(renduMonnaies(pieces,somme-pieces[i]))

        listeSomme =[]

        for i in result:

            listeSomme.append(sommePiece(i))

        var=indiceMin(listeSomme)

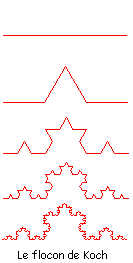
        result = result[var]

        result[var]+=1

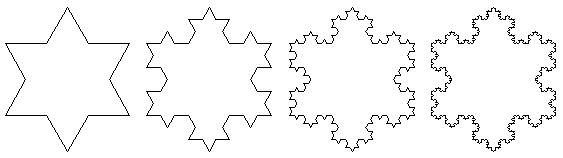
        return result

1. Dessiner le flocon de Koch.

Dessiner sur une ligne.



Et dessiner ensuite tout le flocon.



import turtle

t = turtle.Turtle()

t.speed(0)

turtle.delay()

def koch(longueur, n):

    if n == 0:

        t.forward(longueur)

    else:

        koch(longueur/3, n-1)

        t.left(60)

        koch(longueur/3, n-1)

        t.right(120)

        koch(longueur/3, n-1)

        t.left(60)

        koch(longueur/3, n-1)

def flocon(taille, etape):

    koch(taille, etape)

    t.right(120)

    koch(taille, etape)

    t.right(120)

    koch(taille, etape)

flocon(300, 4)

Pour les plus rapides :

Calendrier de l’avent du code :

https://adventofcode.com/2021/day/1

https://adventofcode.com/2021/day/3

https://adventofcode.com/2021/day/6