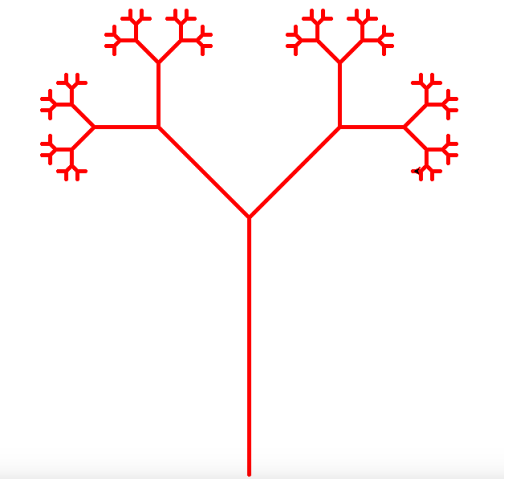
Exercice 1 *Préparation*

1. Copier le sous-dossier `Activite3` du dossier `ICN\Images` situé dans le partage `Donnees` de sa classe. Coller ce dossier dans son espace personnel.
2. `Activite3` contient un sous-dossier `ressources` avec les fichiers images que nous allons manipuler. Ouvrir `Pyzo`, enregistrer dans `ressources` un fichier `Images-Activite3.py`, puis exécuter ce fichier avec l'option `Run as script`.

1 L-système et fractales

Exercice 2



On veut tracer l'arbre binaire ci-contre avec la tortue logo du module `turtle`.
On va d'abord construire une suite de caractères qui va coder les déplacements successifs de la tortue. On part d'une chaîne initiale puis à chaque étape on applique des règles de réécriture : chaque caractère lu dans la chaîne précédente est remplacé par de nouveaux caractères dans la nouvelle chaîne.

- *chaîne initiale* : '0' ;
- *règles de réécriture* : '1' $\mapsto$ '11' et '0' $\mapsto$ '1[0]0'.

La tortue va interpréter ensuite chaque caractère de la chaîne finale qui est le programme de construction de l'arbre :

- si le caractère lu est '0' ou '1' alors la tortue avance ;
- si le caractère lu est '[' alors on mémorise la position et l'angle de la tortue à la fin d'une liste `memoire` avec `append`, puis la tortue tourne à gauche de 45° pour tracer une nouvelle branche à gauche ;
- si le caractère lu est ']' alors on lève le crayon puis on met à jour la position et l'angle de la tortue avec les derniers mémorisés à la fin de la liste `memoire` (utilisation de `pop`), puis la tortue tourne à droite de 45° pour tracer une nouvelle branche à droite.

1. Déterminer à la main la chaînes de caractère obtenue après 3 itérations successives.
2. Compléter le code de la fonction `arbre(n)` ci-dessous qui retourne la chaîne de caractères obtenue après `n` itérations :

```
def arbre(n):
    chaine = '0'
    for k in range(n):
        chaine2 = ''
        for c in chaine:
            if c == '1':
                chaine2 = ...........
```

```
            elif c == '0':
                chaine2 = ...........
            else:
                chaine2 = ...........
        chaine = chaine2
    return chaine
```

3. Importer le module `turtle` puis récupérer la fonction `trace_arbre(n , pas)` dans le fichier `cadeau.py` pour obtenir la figure précédente avec `trace_arbre(6, 10)`.

```
def trace_arbre(n , pas):
    chaine = arbre(n)
    up()
    goto(0,-400)
    setheading(90)
    speed(0)
    down()
    memoire = []
    for c in chaine:
        if c == "0" or c == "1":
            forward(pas)
        elif c == "[":
            memoire.append((heading(), pos()))
            left(45)
        else:
            (angle, position) = memoire.pop()
            up()
            setheading(angle) # mise a jour de l'angle
            setpos(position) # mise a jour de la position
            right(45)
            down()
```

4. Ce modèle de croissance s'appelle un *L-système* et il a été inventé par un biologiste hongrois *Aristid Lindenmayer*, en 1968.

Avec un *L-système* on peut aussi tracer des approximations itérées de *courbes fractales* comme la *courbe du dragon* :

- *chaîne initiale* : 'G' ;
- *règle de réécriture* : la chaîne précédente `chaine` est remplacée par la concaténation `chaine + 'G' + chainemiroir` où `chainemiroir` est obtenue en parcourant `chaine` de droite à gauche et en remplaçant chaque caractère lu 'G' par 'D' et 'D' par 'G'.

La tortue va interpréter ensuite chaque caractère de la chaîne finale :

- avant chaque caractère lu la tortue avance ;
- si le caractère lu est "G" la tortue tourne à gauche de 90° ;
- si le caractère lu est "D" la tortue tourne à gauche de 90°.

a. Définir *une courbe fractale* et préciser la dimension de Hausdorff de la *courbe du dragon* et les propriétés géométriques qui en découlent.

b. Déterminer à la main la chaîne de caractères obtenue après 3 itérations successives et tracer à la main la courbe obtenue.

c. Compléter le code de la fonction `miroir(chaine)` ci-dessous qui retourne la chaîne miroir obtenue en parcourant `chaine` de droite à gauche et en remplaçant 'D' par 'G' et vice-versa :

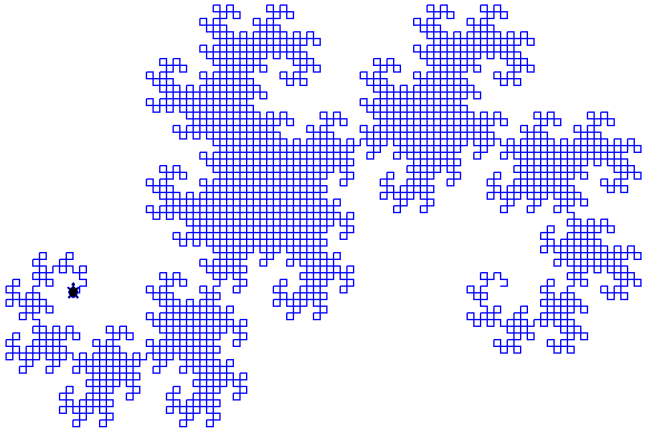
```
def miroir(chaine):
    chaine2 = ''
    for c in chaine:
        #a completer
    return chaine2
```

d. Écrire une fonction `dragon(n)` qui retourne la chaîne de caractères obtenue après `n` itérations.

```
In [78]: miroir('GDG')
Out[78]: 'DGD'

In [79]: dragon(6)
Out[79]: '
     GGDGGDDGGGDDGDDGGGDGGDDDGGDDGDDGGGDGGDDGGGDDGDDDGGDGGDDDGGDDGDD
     '
```

e. Sur le modèle de la fonction `trace_arbre(n , pas)`, écrire une fonction `trace_dragon(n, pas)` qui réalise le tracé d'une approximation de la courbe du dragon en `n` itérations.



2 Stéganographie

Exercice 3 *Codage binaire*

Lire l'article de la page web https://interstices.info/jcms/c_41905/nom-de-code-binaire.

1. Quelle est l'unité élémentaire d'information manipulée par un ordinateur ?

2. Quelle est la définition d'un *byte* (*octet* en français) ?

3. Une image bitmap a pour dimension (Largeur,Hauteur)=(700,800) dont les pixels sont codés en (R,G,B,A) avec une profondeur de 32 bits. La composante Alpha mesure la transparence d'un pixel. Si l'image n'est pas compressée, calculer son poids en mégaoctets puis en mébioctets sans tenir compte de l'entête du fichier.

Bloc-Note

Pour convertir un entier en une série de bits, il suffit de l'écrire en base 2, on parle d'*écriture binaire*.
L'*écriture décimale* de treize est 13 car $1 \times 10 + 3 = 13$.
Son *écriture binaire* est $(1101)_2$ car $1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13$.

- En Python la fonction `bin` retour l'écriture binaire d'un entier sous forme de chaîne de caractère préfixée de '0b' et sa réciproque est la fonction `int` avec le paramètre optionnel 2.

```
In [92]: bin(13)
Out[92]: '0b1101'

In [93]: int('0b1101', 2)
Out[93]: 13
```

- L'opérateur » permet de décaler les bits vers la droite, tandis que « les décale vers la gauche :

```
In [94]: 13 >> 1
Out[94]: 6

In [95]: bin(6)
Out[95]: '0b110'

In [96]: 13 << 1
Out[96]: 26

In [97]: bin(26)
Out[97]: '0b11010'
```

- L'opérateur & permet de réaliser une opération logique ET bit par bit entre deux entiers, tandis que l'opérateur | permet de réaliser une opération logique OU bit par bit entre deux entiers :

Table de vérité du ET

x	y	x ET y
0	0	0
0	1	0
1	0	0
1	1	1

Table de vérité du OU

x	y	x OU y
0	0	0
0	1	1
1	0	1
1	1	1

```
In [101]: bin(13), bin(10)
Out[101]: ('0b1101', '0b1010')

In [102]: 13 & 10, bin(13 & 10)
Out[102]: (8, '0b1000')

In [103]: 13 | 10, bin(13 | 10)
Out[103]: (15, '0b1111')
```

On utilise souvent le ET binaire avec un entier particulier appelé *masque* qu'on applique à un autre entier pour faire apparaître une information par « gommage » des bits qui ne sont pas égaux à 1 comme dans le *masque*. Ce peut être pour récupérer l'adresse IP du sous-réseau dont dépend une machine (voir https://fr.wikipedia.org/wiki/Sous-réseau) ou pour fixer des droits par défaut à tout fichier créé en appliquant le masque à la série 111111111 représentant les droits RWXRWXRWX en lecture (R), écriture (W), exécution (X) pour le propriétaire, le groupe principal ou les autres utilisateurs du fichier.

Exercice 4 *Stéganographie*

La stéganographie consiste à dissimuler un message dans un autre. Par exemple, on veut cacher l'image A `femme-486x624.bmp` dans l'image B `cypres-486x624.bmp`. Ici les images sont de mêmes dimensions mais il suffit que l'image invisible puisse s'intégrer comme un bloc de pixels dans l'image visible. Le format de stockage est important, les formats de compression avec perte comme JPEG ne permettent pas la stéganographie d'images.

Image visible

Image cachée

Image dévoilée

On va créer une image C contenant à la fois les images A et B, et l'apparence de C sera celle de A car les informations extraites de A seront placés au « premier plan » dans le codage numérique de chaque pixel :

☞ Soit a la valeur d'une des trois composantes (R,G,B) d'un pixel de l'image A et b et c les valeur de la même composante du même pixel respectivement dans les images B et C.

a et b sont des entiers compris entre entre 0 et 255 dont la représentation binaire compte 8 bits.

Lorsqu'on lit de gauche à droite une écriture binaire de 8 bits, les premiers bits lus sont dits de *poids forts* et les 3 derniers de *poids faibles*.

Pour construire les 8 bits de c, on prend les 5 de poids forts de a suivis des 3 bits de poids forts de b. On perd de l'information sur a et b mais plus on prend de bits de poids forts moins la perte est importante.

Par exemple avec $a = 166$ d'écriture binaire $a = (\overbrace{10100}^{\text{forts}}110)_2$ et $b = 234$ d'écriture binaire $b = (\underbrace{111}_{\text{forts}}01010)_2$,

on construit $c = (\overbrace{10100}^{a}\underbrace{111}_{b})_2$

- Pour annuler les 3 bits de poids faible de 166 on fait d'abord un ET logique entre 166 et le nombre « masque » d'écriture binaire $(11111000)_2$ qui est 248, on obtient $(10100000)_2$.

– Ensuite on décale de 5 bits vers la droite les bits de l'écriture binaire de 234 avec 234 >> 5 ce qui permet de les obtenir comme bits de poids faibles du nombre d'écriture binaire $(111)_2$ (c'est 7).

– Enfin on effectue un OU logique entre $(\overbrace{10100}\underbrace{000})_2$ et $(\underbrace{111})_2$ et on obtient la réunion de tous ces bits : $c = (\overbrace{10100}\underbrace{111})_2$ avec les trois bits de poids forts de $b = 234$ à la place des trois bits de poids faibles de $a = 166$.

```
In [107]: (a, b) = (166, 234)

In [108]: bin(a), bin(b)
Out[108]: ('0b10100110', '0b11101010')

In [109]: int('0b11111000',2)
Out[109]: 248

In [110]: bin(a & 248)
Out[110]: '0b10100000'

In [111]: bin(b >> 5)
Out[111]: '0b111'

In [112]: c = (a & 248) | (b >> 5)

In [113]: c, bin(c)
Out[113]: (167, '0b10100111')
```

☞ On recommence pour chaque composante et chaque pixel

1. Écrire une fonction `cacher(im1, im2)` qui cache l'image `im2` dans l'image `im1` en appliquant l'algorithme décrit ci-dessus.

```
In [7]: im1 = Image.open('cypres-486x624.bmp')

In [8]: im2 = Image.open('femme-486x624.bmp')

In [9]: im3 = cacher(im1, im2)

In [10]: im3.save('cypres-486x624+femme-486x624.bmp')

In [11]: im4 = devoiler(im3)

In [12]: im4.save('femme-486x624-devoile.bmp')
```

2. Pour transformer les 3 bits de poids faibles d'une composante comme bits de poids forts, on commence par annuler tous les autres bits à gauche en faisant un ET binaire avec $7 = (111)_2$ puis on fait un décalage de 5 bits vers la gauche.

```
In [14]: c, bin(c)
Out[14]: (167, '0b10100111')
```

```
In [15]: bin(c & 7)
Out[15]: '0b111'

In [16]: bin((c & 7) << 5)
Out[16]: '0b11100000'
```

Écrire une fonction `devoiler(im1)` qui retourne l'image cachée par la fonction `cacher`.

3 Enjeux sociétaux et droits du numérique

Exercice 5 *Sujets d'exposés*

Par groupe de trois, traiter l'un des sujets suivants sous forme de présentation numérique (diaporama Impress, page web HTML, prezi ...) que vous devrez présenter en dix minutes devant la classe au cours du premier trimestre.

Pour la réalisation de diaporama avec Impress, la ressource suivante est très bien : https://dane.ac-lyon.fr/spip/Video-Prise-en-main-d-Impress.

- L'art assisté par ordinateur relève-t-il de la création artistique ?

  Ressources : https://interstices.info/jcms/i_58118/l-art-assiste-par-ordinateur

- Watermarking et authentification d'images.

  Ressources :

  https://interstices.info/jcms/c_32093/cryptographie-steganographie-et-tatouage-des-se

  https://fr.wikipedia.org/wiki/Tatouage_numérique

- Modélisation mathématique dans l'imagerie médicale :

  Ressources :

  https://interstices.info/jcms/i_53813/les-mathematiques-cachees-de-la-medecine

  https://interstices.info/jcms/c_12343/christian-barillot-ce-que-pourraient-voir-les-

- Traitement d'image par ordinateur : histoire et champs d'application.

  Ressources :

  https://interstices.info/jcms/c_5952/histoire-du-traitement-d-images.

- Reconnaissance biométrique (contour de la main, empreinte digitale, iris de l'oeil, contour de visage ...) et respect des droits individuels.

  Ressources :

  https://www.cnil.fr/fr/le-controle-dacces-biometrique-sur-les-lieux-de-travail

  http://eduscol.education.fr/internet-responsable/ressources/legamedia/biometrie.html.

- Indexation automatique d'images, bases de données et respect de la vie privée.

  Ressources :

  https://interstices.info/jcms/c_19256/description-et-indexation-automatiques-des-docu