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 Dessiner avec le module `turtle`

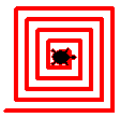
Bloc-Note

Le module `turtle` permet de tracer des figures géométriques en déplaçant une tortue selon une syntaxe proche du légendaire langage `Logo`. Le document `ResumeTurtle.pdf` rassemble les principales instructions, on peut aussi consulter http://www.frederic-junier.org/PythonSeconde/Python_Seconde_Parc/tortue/tortue1.html.

Principales fonctions du module `turtle`

`reset()`	effacer tout et recommencer
`goto(x,y)`	aller au point de coordonnées `(x,y)`
`forward(d)`	avancer de `d` pixels (`d` peut être négatif)
`backward(d)`	reculer de `d`
`up(), down()`	lever ou baisser le crayon
`left(`θ`), right(`θ`)`	tourner à gauche ou à droite de θ degrés
`color('red'),width(1)`	couleur, épaisseur du tracé
`exitonclick()`	ouvre une fenêtre, lance la construction, clic sur la croix pour fermer

Exemple 1 *Un tracé de spirale avec `turtle`*



```
from turtle import *

speed(1)        #parametrage de la vitesse de 1 lent a 10 rapide
shape("turtle") #choix de la forme de la tortue
pencolor("red") #choix de la couleur du crayon
pensize(4)    #epaisseur du crayon
up()          #lever le crayon
goto(0,0)     #aller a la position (0,0)
setheading(0) #orientation de la tortue Est:0/Nord:90/Ouest:180/Sud:270
down()        #poser le crayon
n = 100
while n > 0:
```

```
    for k in range(4):
        forward(n)  #avancer de n pixels
        left(90)    #tourner de 90 degres
        n = n - 5   #reduire n
exitonclick()      #boucle gestionnaire d'evenement
```

Exercice 2

Ouvrir la page http://www.frederic-junier.org/PythonSeconde/Python_Seconde_Parc/tortue/tortue2.html et dans le fichier `Images-Activite3.py` :

- créer une cellule `## Escalier avec turtle` pour traiter l'exercice 1 ;
- créer une cellule `## Polygone  avec turtle` pour traiter l'exercice 2 ;
- créer une cellule `## Spirale  avec turtle` pour traiter l'exercice 3.

2 L-système et fractales

Bloc-Note

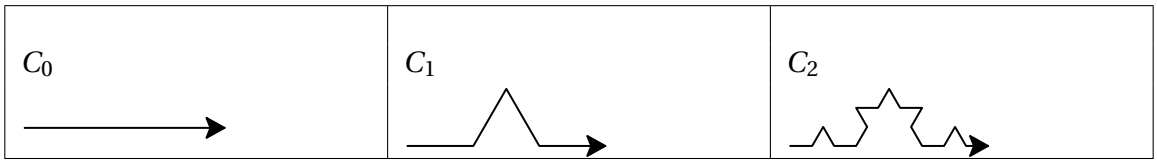
Dans son ouvrage *Les objets fractals – forme, hasard et dimension*, **Benoît Mandelbrot** a introduit la terminologie suivante :

- ☞ Une *courbe fractale* est la limite quand n tend vers l'infini d'une suite de courbes C_n définies récursivement et qui ne font que donner un aperçu de la complexité de la courbe limite.
- ☞ La courbe C_0 est *l'initiateur*.
- ☞ On obtient C_{n+1} à partir de C_n en remplaçant chaque segment composant C_n par un ensemble de segments appelé *générateur*.
- ☞ La courbe C_n est le $n^{\text{ième}}$ *téragone* de la courbe limite.

Exemple 2

Avant la formalisation proposée par **Mandelbrot**, **von Koch** a défini en 1904, une courbe fractale dont l'initiateur est un segment de longueur c et dont le générateur s'obtient en découpant le segment en trois parties de même longueur et en remplaçant le segment médian par deux segments formant avec lui un triangle équilatéral.

On représenté ci-dessous avec `turtle` les trois premiers téragones pour la courbe de **von Koch**.



Exercice 3 *Courbe de Koch*

Les téragones successifs d'une courbe de Koch peuvent être construits à l'aide d'un algorithme inspiré d'un modèle de croissance des plantes appelé L-système qui a été inventé par un biologiste hongrois *Aristid Lindenmayer*, en 1968.

- ☞ On commence par coder les mouvements de la tortue : par exemple `"F"` pour un tracé de segment, `"+"` pour un changement d'orientation de la tortue de 60° vers la gauche et `"-"` pour un changement d'orientation de la tortue de 120° vers la droite.
- ☞ On code le premier téragone par la chaîne de caractères `"F"`.
- ☞ Pour coder le second téragone, on applique le générateur de la courbe de Koch au codage précédent : on remplace `"F"` par la chaîne `"F+F-F+F"`.
- ☞ Pour coder le troisième téragone, on lit le codage du précédent et on remplace `"F"` par le générateur `"F+F-F+F"` et on laisse inchangés les caractères `"+"` et `"-"` de rotation de la tortue.
- ☞ On réitère le procédé jusqu'au téragone ciblé.
- ☞ Ensuite on lit la chaîne de caractères codant le tracé du téragone et on traduit chaque caractère par un déplacement de la tortue selon le codage préétabli.

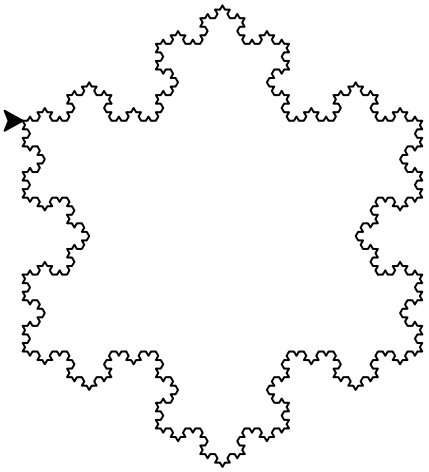
1. Déterminer à la main les chaînes de caractère codant les troisième et quatrième téragones de la courbe de Koch.

2. Compléter le code de la fonction `koch(n)` ci-dessous qui retourne la chaîne de caractères obtenue après `n` itérations :

```
def koch(n):
    code = "F"
    for k in range(n):
        code2 = ""
        for c in code:
            if c == "F":
                "completer"
            else:
                "completer"
        code = code2
    return code
```

3. Importer le module `turtle` puis écrire une fonction `trace_koch(n, pas)` qui trace le $n^{\text{ième}}$ de la courbe de Koch.

4. Écrire une fonction `trace_flocon_koch(n, pas)` qui trace le flocon de Koch obtenu en combinant trois téragones de même ordre de la courbe de Koch.



Exercice 4 *Courbe du dragon*

Avec un *L-système* on peut tracer des téragones d'autres *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 droite de 90°.

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

2. 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 :

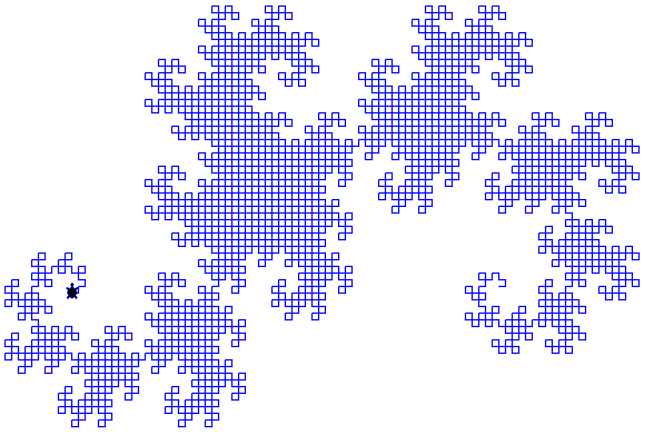
```
def miroir(chaine):
    chaine2 = ''
    for c in chaine:
        #a completer
    return chaine2
```

3. 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'
```

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



3 Stéganographie

Exercice 5 *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) et ses 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 6 *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 placées 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 derniers de *poids faibles*.

Pour construire les 8 bits de c, on prend les 5 bits de poids forts de a suivis des 3 bits de poids forts de b. On perd de l'information sur a et b : plus on prend de bits de poids forts moins la perte est importante pour b mais plus l'apparence de l'image C s'éloigne de celle de A.

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`.

4 Enjeux sociétaux et droits du numérique

Exercice 7 *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.

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