Tous les exercices sont précédés de la consigne ci-après : *Cet exercice est prévu pour le langage Python (et ses bibliothèques numpy, scipy, matplotlib). À chaque question, les instructions ou les fonctions écrites devront être testées.*

Exercice 1

1. Définir trois points $A(28,5)$, $B(13,27)$, $C(2,2)$, puis faire tracer le triangle ABC.

2. Soit la suite récurrente de points $(T_n)_{n\in\mathbb{N}}$ définie par : $T_0 = D$ (fixé); T_{n+1} est le milieu de T_n et d'un point tiré aléatoirement parmi A, B et C (équiprobabilité).
   Écrire une fonction `suivant` d'argument un point T et qui renvoie le point S, milieu de T et d'un point tiré aléatoirement parmi A, B et C.
   On pourra utiliser la fonction `rand`, du module `numpy.random` de *Python*, qui tire un nombre au hasard dans l'intervalle $[0,1]$.

3. Pour $D(1,27)$, construire une matrice (11×2) contenant les abscisses et les ordonnées de T_n, pour n variant de 0 à 10. Tracer le nuage de points correspondant.

4. Tracer les 10000 premiers points de la suite (T_n).

5. Définir la fonction `PT` de cinq arguments A, B, C, D et N qui renvoie les N premiers points de suite $(T_n)_{n\in\mathbb{N}}$, sous la forme d'une matrice $(N\times 2)$.
   Utiliser cette fonction pour tracer les 10000 premiers points de la suite pour des points A, B, C et D tirés au hasard dans le carré $[0,30]\times[0,30]$.

Exercice 2 On se place dans le plan muni d'un repère orthonormé. Chaque point du plan est caractérisé par une liste de deux réels du type `x,y`.
On rappelle que, si C est un point et r un réel, l'image M' du point M par l'homothétie de centre C et de rapport r est l'unique point M' tel que :

$$\overrightarrow{CM'} = r\overrightarrow{CM}.$$

1. Écrire une fonction `H` de trois arguments, un point C, une valeur réelle r et un point M qui renvoie l'image de M par l'homothétie de centre C et de rapport r.

On se donne m homothéties du plan de centres C_i et de rapports $r_i \in]0,1[$.
On appelle *jeu du chaos* la suite de points $(M_k)_{k\geqslant 0}$ construite de la manière suivante :

- le point M_0 est donné ;
- pour tout $k \geqslant 0$, on pose $M_{k+1} =$`H`$(C_i, r_i M_k)$ où i est tiré au sort à chaque itération avec une loi équiprobable. Avec *Python*, on pourra utiliser la fonction `randint` de la bibliothèque `random`.

2. Écrire une fonction `tirer` de quatre arguments, la liste des centres des homothéties, la liste des rapports des homothéties, le point M_0 et le nombre n d'itérations qui renvoie la liste des $n+1$ premiers points de la suite.

3. Tracer les 5001 premiers points de la suite définie par les trois homothéties de rapports 0.5 et de centres trois sommets d'un triangle équilatéral et de premier point l'un de ces sommets.

4. Définir une fonction `T` de trois arguments m, r et n qui trace les $n+1$ premiers points de la suite obtenue pour un jeu d'homothéties de même rapport r et dont les centres sont les sommets d'un polynôme régulier à m côtés et de premier point l'un de ces sommets.
   Tester `T(5,0.37,10000)`.

Exercice 3

1. Définir une fonction f d'argument x qui renvoie $1 + \lfloor 1/x \rfloor$, où $\lfloor t \rfloor$ désigne la *partie entière de* t (`floor`), si x est strictement positif, et zéro sinon.

Á toute liste $B = [b_1, b_2, \cdots, b_\ell]$ de ℓ valeurs non nulles ($\ell \geqslant 2$), on associe :

$$S(B) = \frac{1}{b_1}\left(1 + \frac{1}{b_2}\left(1 + \cdots \left(1 + \frac{1}{b_\ell}\right)\cdots\right)\right).$$

Par exemple, $S([1,3,2]) = \dfrac{1}{1}\left(1 + \dfrac{1}{3}\left(1 + \dfrac{1}{2}\right)\right) = \dfrac{3}{2}$.

On pose également $S([b_1]) = \dfrac{1}{b_1}$.

2. En remarquant que $S(B)$ s'écrit $\dfrac{1}{b_1}(1 + S(B'))$, définir la fonction S.
   Tester cette fonction sur la liste `[1,3,2]`.

3. À partir d'un nombre x strictement positif, on définit la suite (a_n) par :

   - $a_0 = f(x)$.
   - $\forall n \in \mathbb{N}^*,\ a_n = f(a_0 a_1 \cdots a_{n-1}(x - S([a_0, a_1, \cdots, a_{n-1}])))$.

   Écrire la fonction `LA` de deux arguments x et n, qui renvoie les $n+1$ premiers termes de la suite (a_n) en partant de x.

4. Tester `LA(1,5)`, puis `S(LA(1,5))`.
   Faire de même pour `LA(5/7,4)`, puis `S(LA(5/7,4))`.
   Tester `LA(5/7,5)`. Commenter.

5. Écrire une fonction `Srec` de deux arguments x et d qui renvoie la première liste $[a_0, a_1, \cdots, a_\ell]$ trouvée telle que $x - S([a_0, a_1, \cdots, a_\ell]) \leqslant 10^{-d}$. Tester cette fonction pour $x = 1$ et $d = 7$.

Exercice 4 Une année est bissextile si elle est divisible par 4 mais pas par 100, ou si elle est divisible par 400.

1. *Question préliminaire* : soit $n = 4321$. Comment obtient-on le reste de la division euclidienne de n par 25 ? Comment obtient-on le quotient ?

2. Écrire une fonction `Bissextile` d'argument a et renvoyant un booléen, déterminant si une année a est bissextile ou pas. La tester avec les dates 1900, 1995, 1996, 2000.

3. Écrire une fonction `NumeroJour` de trois arguments, le jour `j`, le mois `m`, et l'année `a`, renvoyant le numéro du jour dans l'année `a`, compris entre 1 et 365 ou 366. Tester votre procédure avec le 5 Juillet 2015 (réponse : 186).

4. Écrire une fonction `NombreJours` de trois arguments, le jour `j`, le mois `m`, et l'année `a`, renvoyant le nombre de jours écoulés depuis le premier janvier 1900. Tester votre procédure avec le 5 Juillet 2015 (réponse : 42189).

5. En informatique, un problème similaire au bogue de l'an 2000 pourrait perturber le fonctionnement d'ordinateurs 32 bits. Le problème concerne des logiciels qui utilise la représentation POSIX du temps, dans lequel le temps est représenté comme le nombre de secondes écoulées depuis le premier Janvier 1970, 0 : 00 : 00. Sur les ordinateurs 32 bits, la plupart des systèmes d'exploitation

concernés codent ce nombre comme un nombre entier signé de 32 bits, ce qui limite le nombre de secondes à $2^{31} - 1$.
Déterminer en quelle année pourrait se produire le bogue de la représentation POSIX sur un ordinateur 32 bits, s'il en existe encore.

Exercice 5 Soit Ω_V le sous-ensemble $\{(1,0),(-1,0),(0,1),(0,-1)\}$ de l'espace vectoriel $\mathbb{R}^2$.
Soit (V_n) une suite de variables aléatoires à valeurs indépendantes suivant la même loi uniforme sur Ω_V.
On définit une suite de variables aléatoires (X_n) en posant :

$$X_0 = (0,0), \quad \text{et} \quad \forall n \in \mathbb{N}, \quad X_{n+1} = X_n + V_n.$$

On appelle "trajectoire de longueur n" la ligne brisée aléatoire reliant les points de coordonnées $X_0, X_1, \cdots, X_n$.

1. Écrire une fonction `tirage` sans argument et renvoyant une des listes suivantes avec équiprobabilité : `[1,0]`, `[-1,0]`, `[0,1]` et `[0,-1]`.
   On pourra utiliser la fonction `rand` du module `numpy.random` de *Python*.

2. En déduire une fonction `trajectoire` d'argument n renvoyant le tirage d'une trajectoire de longueur n, sous forme d'une liste de couples `[x,y]`.

3. Tracer quelques trajectoires de longueurs 10, 100, 1000, puis 10000.

4. Écrire une fonction `premierRetour` d'argument m renvoyant le plus petit entier non nul $n \leqslant m$ tel que $X_n = (0,0)$ s'il existe, et -1 sinon.

5. On note T le premier retour en $(0,0)$. Au moyen d'un programme, conjecture si la variable aléatoire T est bien définie.

Exercice 6

1. Que fait la fonction `L2str` suivante ? L'appliquer à la liste `[1,2,1,1]`.

```
def L2str(L) :
    ch = ""
    for e in L :
        ch = ch + str(e)
    return ch
```

Soit une suite d'entiers naturels $(u_n)_{n\in\mathbb{N}}$ définie par : $u_0 = 1$ et $\forall n \in \mathbb{N}$, $u_{n+1} = \phi(u_n)$, où la fonction ϕ est définie comme suit : $\phi(k)$ est le nombre composé de chiffres décrivant le nombre k. Par exemple :

1112	$\mapsto$	3112	(trois uns, un deux)
29	$\mapsto$	1219	(un deux, un neuf)
333	$\mapsto$	33	(trois trois)
1211	$\mapsto$	111221	(un un, un deux, deux uns)

Dans cet exercice, chaque entier naturel sera codé sous forme d'une liste de chiffres.

2. Écrire une fonction `lire` d'argument une liste L de chiffres codant un nombre k et renvoyant la liste des chiffres de $\phi(k)$. Par exemple, `lire([1,2,1,1])` doit donner `[1,1,1,2,2,1]`.

3. Afficher les quinze premiers termes de la suite u_n.

4. Quels sont les chiffres présents dans u_n ? Expliquer succinctement pourquoi.

5. Pour tout entier naturel n, on note ℓ_n le nombre de chiffres de u_n. On peut montrer l'existence de k et λ dans $\mathbb{R}_+^*$ tels que $\ell_n \underset{n\to+\infty}{\sim} k\lambda^n$.
   Vérifier numériquement ce résultat en donnant des valeurs grossièrement estimées de k et de λ.

6. Étudier la proportion des différents chiffres dans u_n.

Exercice 7 On cherche à étudier numériquement les solutions de l'équation différentielle avec conditions initiales suivante

$$y'(t) = t^2 - y(t)^3 \quad \text{avec} \quad y(-1.5) = a. \tag{1}$$

1. Pour $a = 2$ et un pas de discrétisation $h = \frac{1}{4}$, puis $h = \frac{1}{8}$, représenter les solutions approchées du problème (1) obtenues par la méthode d'Euler sur l'intervalle $[-1.5, 2.5]$.

2. En utilisant la fonction `odeint` du module `scipy.integrate` de *Python*, résoudre numériquement le problème (1) pour $a = 2$. On prendra garde à bien définir soigneusement les arguments de cette fonction en lisant attentivement l'aide en ligne.

3. Sur la figure existante, rajouter la courbe de la solution numérique obtenue à la question précédente.

4. Rajouter ensuite les courbes des solutions numériques pour $a = 1.3$ et $a = 0.1$. Qu'observe-t-on ?

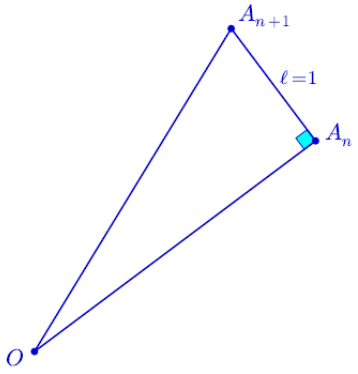
5. Pour résoudre $y'(t) = f(t, y(t))$ avec $y(t_0) = a$, on utilise maintenant la suite (y_n) définie par $y_0 = a$ et $y_{k+1} = y_k + \dfrac{h}{2}\left(f(t_k, y_k) + f(t_{k+1}, y_k + hf(t_k, y_k))\right)$.
   Représenter, pour $a = 2$ et les mêmes pas de discrétisation qu'à la question 1, les solutions approchées obtenues par cette méthode. Expliquer pourquoi le résultat semble meilleur.

Exercice 8 Dans le plan affine, muni d'un repère orthonormé direct d'origine O, on considère la suite de point $(A_n)_{n\in\mathbb{N}}$ tels que :

- A_0 a pour coordonnées $(1, 0)$.
- pour tout entier naturel non nul n, le triangle OA_nA_{n+1} est rectangle en A_n, la distance A_nA_{n+1} vaut 1 et l'angle $\left(\overrightarrow{OA_n}, \overrightarrow{OA_{n+1}}\right)$ est direct.

Si x_n et y_n sont les coordonnées de A_n, on a :

$$\begin{cases} x_{n+1} = x_n - \dfrac{y_n}{\sqrt{x_n^2 + y_n^2}} \\ y_{n+1} = y_n + \dfrac{x_n}{\sqrt{x_n^2 + y_n^2}} \end{cases}.$$



1. Écrire une fonction `A` d'argument N renvoyant la liste des coordonnées des points A_n pour $0 \leqslant n \leqslant N$.

2. Écrire une fonction `afficher` d'argument N affichant la figure représentant les $(N + 1)$ premiers points A_n. La tester pour $N = 60$.

Les points A_k tournent autour de l'origine O du repère. On note $T(k)$ la valeur de n telle que le point A_n commence le k-ième tour.

3. Écrire une fonction `tours` d'argument m qui renvoie la liste des $T(k)$ pour k variant de 1 à m. Afficher `tours(5)`.

4. Faire tracer chacun des cinq premiers tours avec une couleur différente.

5. Déterminer les abscisses des dix premiers points d'intersection de la ligne reliant les A_n avec la partie positive de l'axe des abscisses. Vérifier qu'à chaque tour, on s'est éloigné du centre d'une distance environ égale à π.

Exercice 9 Une méthode pour estimer numériquement l'aire du disque de centre O et de rayon 1 est la suivante : on tire au hasard N points de coordonnées $(x, y) \in [-1, 1] \times [-1, 1]$; parmi ces points, on compte le nombre i de ceux qui appartiennent au disque ; On admet que i/N est une approximation du rapport de l'aire du disque par l'aire du carré.

1. Observer et expliquer ce que fait le code suivant :

```
1 from numpy.random import rand
2 Lpts = 2*rand(8,2) - 1
3 print(Lpts)
```

2. Écrire une fonction `estim1` d'argument N qui renvoie une valeur approchée de l'aire du disque, calculée selon le procédé indiqué ci-dessus.

3. Écrire une fonction `estim2` analogue à `estim1` qui fait tracer en plus les points tirés dans le carré ; ceux dans le disque avec une couleur, et ceux à l'extérieur avec une autre couleur.
Pour obtenir un repère orthonormé, on pourra utiliser l'instruction `axis(''equal'')` grâce au module `matplotlib.pyplot` de *Python*.

On s'intéresse maintenant à l'aire du domaine $\mathcal{D}$ d'équation : $(x^2 + y^2)^2 \leqslant x^3$.

4. Faire tracer l'allure de la frontière du domaine $\mathcal{D}$ après avoir résolu "*à la main*" en y dans $\mathbb{R}$ l'équation $(x^2 + y^2)^2 = x^3$.

5. En déduire que le domaine $\mathcal{D}$ est contenu dans un rectangle que l'on déterminera.

6. À l'aide de la méthode précédente, estimez l'aire du domaine $\mathcal{D}$ et faire afficher avec des couleurs différentes les points tirés, selon qu'ils seront dans $\mathcal{D}$ ou pas.