Programmation sous `python` — 2025-2026

Contrôle continue n° 1
11 février 2026, 13:30–15:00

À lire. Quelques consignes importantes pour la notation :

- Les scripts doivent être commentés et syntaxiquement corrects : un script qui génère une erreur de syntaxe est systématiquement sanctionné (pas de "petite croix rouge" sous Spyder !)
- Internet est autorisé ainsi que vos travaux en séances. Les échanges de mails et l' utilisation d'intelligences artificielles (type ChatGPT) sont interdits.
- Le non-respect des consignes est aussi pénalisé (si on demande une fonction `somme(n)`, on ne veut pas trouver une fonction `SOMME(n)` ou encore `Somme(N)` comme réponse).
- Votre script `cc1_26.py` est à déposer sur Moodle dans l'espace dédié à votre groupe dans `UE 4 : Programmation sous Python`. (Chercher le nom de l'enseignant !)
- Le barème est sur 21 : il n'est pas nécessaire de tout faire pour avoir la note maximale.

Exercice 1. Les questions de cet exercice sont indépendantes

1) Écrire une fonction Python `estDansLaBoule` qui prend en argument un vecteur $x \in \mathbb{R}^3$ et un rayon $r > 0$ et qui renvoie le booléen `True` ou `False` indiquant si x appartient ou non à la boule (fermée) de rayon r centrée à l'origine, $B(r) = \{x \mid x_1^2 + x_2^2 + x_3^2 \leq r^2\}$.

Tester cette fonction pour

- $r = 0.5937$ et $x = (-\frac{1}{2}, -\frac{1}{4}, \frac{1}{5})$
- $r = 0.3$ et $x = (0.2, 0.2, 0.1)$
- $r = 3$ et $x = (2, 2, 1)$.

Quelle conclusion peut-on tirer des deux derniers exemples ? Rédiger la réponse en utilisant la commande `print`.

2) Écrire une fonction Python `melanger` qui prend en argument deux chaînes de caractères x et y et renvoie la chaîne

$$x[0]y[0]x[1]y[1]\dots$$

Par exemple

$$\text{'abcd', 'xy'} \longmapsto \text{'axbycd'}.$$

Tester la fonction pour

- $x =$ 'a' et $y =$ 'bc'
- $x =$ 'ab' et $y =$ 'c'
- $x =$ 'abcd' et $y =$ 'd ef'.

3) Soit $M \in \mathcal{M}_n(\mathbb{R})$ une matrice réelle carrée d'ordre n. On rappelle que toute matrice M peut se décomposer de manière unique sous la forme

$$M = S + A = \frac{1}{2}(M + M^\top) + \frac{1}{2}(M - M^\top)$$

où S est symétrique et A antisymétrique (c'est-à-dire $S^\top = S$ et $A^\top = -A$). Écrire une fonction Python `symEtAntisym` qui prend en argument un tableau `numpy` de dimension deux carré et qui renvoie les tableaux S, A, P, où P est le produit matriciel S^n.

4) Écrire les instructions nécessaires pour dessiner un cercle de rayon 2 centré en $(0, 0)$ ainsi que deux de ces diamètres qui font un angle de 45 degrés entre eux.

Solution.

1) Une possibilité pour la fonction `estDansLaBoule` est donnée ci-dessous.

```python
def estDansLaBoule(x, r):
    """
    x=(x1,x2,x3) est une liste ou vecteur
    r est le rayon de la boule centre a l'origine
    """
    x1, x2, x3 = x
    tmp = x1**2 + x2**2 + x3**2
    return tmp <= r**2
```

Sur les trois tests demandés, les résultats sont False, False et True respectivement. On remarque que les deux derniers tests diffèrent seulement par un facteur 10. Le résultat devrait être le même. La différence est due au fait que l'égalité avec 0 n'a pas de sens en virgule flotante ; les deux points se trouvent sur la frontière de la boule.

2)

```python
def melanger(x, y):
    res = ''
    n = min(len(x), len(y))
    for k in range(n):
      res += x[k] + y[k]
    if len(x) > len(y):
      res += x[n:]
    elif len(x) < len(y):
      res += y[n:]
    return res
```

3)

```python
def symEtAntisym(M):
    """
    M est une matrice carree.
    """
    S = 0.5*(M + M.T)
    A = 0.5*(M - M.T)

    n, m = np.shape(M)
    P = np.eye(n)
    for k in range(n):
        P = P.dot(S)
    return S, A, P
```

4)

```python
a = np.sqrt(2)
theta = np.linspace(0, 2*math.pi, 200)
x_cercle = 2*np.cos(theta)
y_cercle = 2*np.sin(theta)

plt.figure()
plt.title("un cercle rayon 2 et deux diametres a 45 degres")
plt.axis("equal")
plt.grid(True)
plt.plot(x_cercle, y_cercle, color='k')
plt.plot([-2, 2], [0, 0], color='b', linewidth=.5)  # un diametre horizontal
plt.plot([-a, a], [-a, a], color='r', linewidth=.5)  # un diametre a 45
plt.show()
```

Exercice 2. On considère la série numérique

$$S = \sum_{n\geq 0} (-1)^n \frac{1}{(2n+1)3^n}\,.$$

On rappelle que la somme partielle S_N est définie par

$$S_N = \sum_{n=0}^{N} (-1)^n \frac{1}{(2n+1)3^n}\,.$$

1) Écrire une fonction `sommePartielle` prenant en argument un entier naturel n et renvoyant S_n, la n-ème somme partielle de la série.

2) Écrire une fonction `sommesPartielles` prenant en argument un entier naturel n et renvoyant la liste des premières $n+1$ sommes partielles, $[S_0, S_1, \ldots, S_n]$.

3) En utilisant la bibliothèque `matplotlib`, écrire une fonction `evolutionS` prenant en argument un entier naturel n et ayant comme effet de bord la représentation graphique des $n+1$ premiers termes de la suite des sommes partielles de la série. Le système de coordonnées aura les indices $\leq n$ an abscisse et le dessin aura *Premieres n sommes partielles* comme titre, où n est l'argument de la fonction.

4) Tester `evolutionS` pour $n = 10^2$. Conjecturer le comportement asymptotique de la suite des sommes partielles de la série, c'est-à-dire une valeur approchée $\widetilde{S}$ pour S, la somme de la série. Indiquer $\widetilde{S}$

- en utilisant une commande `print` à la fin de la fonction `evolutionS`, et
- en rajoutant à la figure précédente la droite $y = \widetilde{S}$ en rouge.

5*) Écrire une fonction `calculApproche` qui prend en argument un entier strictement positif d et qui renvoie un N tel que

$$|S - S_N| < \frac{1}{10^d}.$$

On pourra utiliser une borne pour la différence $S - S_N$ donner par le critère de convergence des séries alternées.

Solution.

1) La fonction `sommePartielle` est simmilaie à celle développée pour certains exercices de la feuille no 2.

```
def sommePartielle(n):
    S = 0
    for k in range(n):
        S += (-1/3)**k / (2*k + 1)
    return S
```

2) Il est plus sain de construire la liste sans appeler la fonction `sommePartielle`.

```
def sommesPartielles(n):
    L = []
    S = 0
    for k in range(n):
        S += (-1/3)**k / (2*k + 1)
        L.append(S)
    return L
```

En regardant la sortie pour `sommePartielle(10)`, on remarque que la somme de la série sera proche de 0.9068995.

3)

```
n = 25
stilde = 0.9068995
x = list(range(n + 1))
```

```
    y = sommesPartielles(n)
    plt.figure()
    plt.title(f"Premieres {n} sommes partielles")
    # plt.axis("equal")
    plt.grid(True)
    plt.plot(x, y, color='b', lw=2, marker='o', ms='5')
    plt.plot([0, n], [stilde, stilde], color='r', lw=.5)
    plt.show()
```

4) Voir la question 2).

5) D'après le critère des séries alternées,

$$|S - S_N| \leq u_{N+1} = \frac{1}{(2N+3)3^{N+1}}.$$

Il s'ensuit qu'on cherche le premier N tel que

$$\frac{1}{(2N+3)3^{N+1}} < \frac{1}{10^d}.$$

```
def terme(n):
    return 1/3**n / (2*n + 1)


def calculApproche(d):
    e = 10**(-d)
    N = 1
    while terme(N + 1) > e:
        N = N + 1
    return sommePartielle(N), N
```

On remarque que la valeur approchée à 10^{-10} près est 0.906899682. Le $N = 17$ n'est probablement pas le premier N pour lequel

Barème indicatif : 10 — 11