Calcul scientifique avec `python` – 2024-2025

Contrôle continu
26 mai 2025 — 9:30-11:30

Sont autorisés seulement vos documents, fichiers personnels et accès à l'internet, sauf l'email. Le travail informatique sera réalisé et les réponses éventuelles données au choix

- dans un unique fichier `exam.py`
- dans des fichiers nommés `exo1.py`, `exo2.py`...

Les fichiers sont à déposer dans l'espace correspondant sur Moodle sur la page

`UE 4 : Programmation sous Python`

1. Moodle rajoute votre nom au fichier automatiquement.
2. Il faudrait que, lors de la compilation du fichier, les résultats de chacun des exercices apparaissent sous forme de `print(...)` ou dessin. Les mots explicatifs seront appréciés.

   En claire, ne commentez pas avec des `#` les tests effectués !
3. Chaque question vaut 2 points sauf si elle est *).

   Les questions *) ne sont pas plus difficiles que les autres. Par contre, elles ne demandent pas du code, seulement de la réflexion (sur papier). Chacune vaut 1 point.

Exercice 1. On considère le polynôme $P = X^3 + Y^3 + Z^3 - 31$. On veut montrer que l'équation n'admet aucune solution $(a, b, c) \in \mathbb{Z}^3$.

1) Écrire une fonction `polynome(x, y, z)` qui renvoie la valeur $P(x, y, z)$.

2) Écrire une fonction `solutions_modulo(N)` qui prend en argument un nombre naturel N non nul et renvoie la liste des triplets (a, b, c) avec $a, b, c \in \{0, 1, \dots, N-1\}$, tels que $P(a, b, c) = 0 \mod N$, c'est-à-dire la liste des solutions de $P = 0$ modulo N.

3) Tester la fonction précédente pour $2 \leq N \leq 17$.

4*) Conclure.

Exercice 2. Soit $(a_n)_{n\in\mathbb{N}}$ la suite réelle définie par $a_0 = a_1 = 1$ et telle que pour tout $n \geq 2$,

$$a_n = a_{n-1} + a_{n-2}. \qquad (*)$$

1) Déterminer le premier n naturel pour lequel $a_n \geq 10000$.

2) Écrire une fonction `quotients(n)` qui dessine les termes $q_k = \frac{a_k}{a_{k-1}}$, en fonction de k pour $k \in \{1, 2, \dots, n\}$. Appliquer la pour $n = 100$.

3)

a) Conjecturer une valeur limite q pour la suite $(q_n)_{n\geq 1}$.

b) Tracer, en utilisant une autre figure, les points (k, a_k) et (k, q^k) pour $k \in \{1, 2, \dots, n\}$. (Par exemple, on peut prendre $n = 100$.)

4*) En utilisant la formule $(*)$, c'est-à-dire la définition de a_n, trouver la valeur exact de q. En déduire un équivalent de a_n en $+\infty$. (On pourra vérifier l'équivalent en l'illustrant.)

Exercice 3. Soit la suite $(p_n)_{n\in\mathbb{N}}$ de points dans le plan définie par les relations de récurrence

$$\begin{cases} x_{n+1} = 1 - ax_n^2 + by_n \\ y_{n+1} = x_n \end{cases} \tag{♯}$$

où $p_n = (x_n, y_n)$. Par la suite, on choisira $a = 1.4$ et $b = 0.3$.

1) Écrire une fonction `terme_suivant_H(x0, y0)` qui prend en arguments les coordonnées `x0` et `y0` d'un point et qui renvoie les coordonnées `x` et `y` du terme suivant de la suite (♯).

2) Écrire une fonction `suite_H(x0, y0, N)` qui prend en arguments les coordonnées `x0` et `y0` du terme initial p_0 ainsi qu'un entier `N` et qui renvoie la liste $[p_0, p_1, \ldots, p_N]$. Tester les fonctions en prenant par exemple $x_0 = 0$, $y_0 = 1$ et $N = 2$.

3) Écrire une fonction `esquisse_suite_H(x0, y0, N)` qui prend en arguments les coordonnées `x0` et `y0` du terme initial p_0 ainsi qu'un entier `N` et qui représente les $N + 1$ points de la suite avec un quadrillage de fond, des noms aux axes et un titre adapté. On fera des tests en prenant $N = 500$ (ou 1000) et $p_0 = (0.1, 0.1)$ puis $p_0 = (1.1, -0.2)$. Indiquer en commentaire ce que l'on peut remarquer en inspectant les deux graphiques obtenus.

4) Écrire une fonction `terme_proche_H(x0, y0, xM, yM, eps)` qui prend en arguments :

- `x0`, `y0` : les coordonnées du point initial p_0,
- `xM`, `yM` : les coordonnées d'un point $M = (x_M, y_M)$ quelconque du plan,
- `eps` : un réel strictement positif (en général petit),

et qui renvoie les coordonnées de p_k et l'entier k tels que p_k est le premier terme de la suite pour lequel la distance de p_k à M est plus petite que `eps`.

5) Si $p_0 = (0.1, 0.1)$, déterminer le premier p_k tel que la distance de p_k à $(0.12, 0.8)$ soit plus petite que 10^{-2}. Combien vaut k ? Pouvez-vous déterminer le deuxième terme de la suite qui satisfait ces conditions.

Barème indicatif : 7 — 7 — 10

Solutions

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import (MultipleLocator, AutoMinorLocator)


print("Exercice no 1\n")
def f(x, y, z):
    return z**3 + y**3 + z**3 - 31


def soluitons_modulo(N):
    L = []
    for x in range(N):
        for y in range(N):
            for z in range(N):
                s = f(x, y, z) % N
                if s == 0:
                    L.append([x, y, z])
    return L

```

```
for N in range(2, 11):
    L = soluitons_modulo(N)
    print(f"Le nombre de solutions modulo {N} est {len(L)}.")


print("\nComme il n'y a pas de solution modulo 9, on en deduit\n", \
      "qu'il n'y a pas de solution entiere.\n\n")



print("Exercice no 2\n")

def min_N_fibonacci(bound):
    a = 1
    b = 1
    N = 1
    while b < bound:
        b, a = a + b, b
        N += 1
    return N, b


def fibonacci(n):
    """
    Renvoie la liste des termes a_0, ..., a_n de la suite de Fibonacci.
    """
    L = [1, 1]
    for k in range(2, n + 1):
        a = L[-2] + L[-1]
        L.append(a)
    return L


def quotients(n):
    L = fibonacci(n)
    N = len(L)
    Q = [L[k]/L[k - 1] for k in range(1, N)]
    return Q


n = 10
bound = 17
L = fibonacci(n)
for k in range(n):
    print(f"le terme d'indice {k} de la suite de Fibonacci est {L[k]}")


indice, terme = min_N_fibonacci(bound)
print(f"\nLe premier terme plus grand que {bound} est {terme} d'indice {indice}.")

bound = 10000
indice, terme = min_N_fibonacci(bound)
print(f"\nLe premier terme plus grand que {bound} est {terme} d'indice {indice}.")
print("\n\n")


N = 21
L = fibonacci(N)[1:]  # pour avoir la meme longueur
Q = quotients(N)
X = [k for k in range(1, len(Q) + 1)]


fig = plt.figure(figsize=(14, 6))
```

```
axe_1 = fig.add_subplot(1, 3, 1)
axe_2 = fig.add_subplot(1, 3, 2)
axe_3 = fig.add_subplot(1, 3, 3)

axe_1.set_title("Les quotients pour la\n suite de Fibonacci")
axe_2.set_title("Les comparaisons avec les puissances")
axe_3.set_title("L'equivalent")

axe_1.plot(X, Q, label="quotients")
for q in [1.615, 1.62]:
    axe_1.plot([1, N], [q, q], ls='dashed', lw=.6, label=f"q={q}")
axe_1.legend()


axe_2.plot(X, L, label="fibonacci")
for q in [1.615, 1.62]:
    axe_2.plot(X, [q**k for k in X], ls='dashed', lw=.6, label=f"q={q}")
axe_2.legend()


print("La suite de Fibonacci se comporte comme une puissance.")
print("En utilisant la relation de recurrence, la puissance est" +\
      "\nracine de X**2 -X -1 = 0.")
q1 = (1 +np.sqrt(5))/2
q2 = (1 -np.sqrt(5))/2
print("On en deduit que les termes de la suite de Fibonacci sont des" +\
      f"\ncombinaisons lineaires des puissances de {q1} et de {q2}.")


def terme_fibonacci_explicite(n):
    """
    Les constantes a et b sont les solutions du systeme lineaire
    a + b = 1 et a - b = 1/sqrt(5).

    Renvoie le terme d'indice n calcule comme nombre reel !
    """
    a = 1/2*(1 + 1/np.sqrt(5))
    b = 1/2*(1 - 1/np.sqrt(5))
    q1 = (1 +np.sqrt(5))/2
    q2 = (1 -np.sqrt(5))/2
    return a*q1**n + b*q2**n

n = 7
print(f"\nLe terme d'incide {n} donne par la formule explicite est",
      f"\n{terme_fibonacci_explicite(n)}.\n\n")


Y = [L[k-1]/(1/2*(1 + 1/np.sqrt(5)) * q1**k) for k in X]
axe_3.plot(X, Y)



print("Exercice no 3\n")

def terme_suivant_H(x0, y0):
    a = 1.4
    b = 0.3
    x = 1 - a*x0**2 + b*y0
    y = x0
    return x, y


def suite_H(x0, y0, N):
```

```
    a = 1.4
    b = 0.3
    p = [x0, y0]
    H = [p]
    for k in range(1, N + 1):
        x, y = H[-1]
        p = [1 - a*x**2 + b*y, x]
        H.append(p)
    return H


def premier_terme_proche(p0, point, distance):
    q = point
    p = p0
    n = 1
    d = np.sqrt((p[0] - q[0])**2 + (p[1] - q[1])**2)
    while d > distance:
        p = terme_suivant_H(*p)
        n += 1
        d = np.sqrt((p[0] - q[0])**2 + (p[1] - q[1])**2)
    return n , p


x0 = 0
y0 = 1
N = 3

print(f"Pour x0 = {x0} et y0 = {y0}, le point d'indice 1 a les coordonnees {
    terme_suivant_H(x0, y0)}.")

print(f"Pour x0 = {x0} et y0 = {y0}, les points d'indice <= {N} sont {suite_H(x0, y0
    , N)}.")


N = 500
p0_1 = [.1, .1]
p0_2 = [1.1, -.2]
point = [.12, .8]

H1 = suite_H(*p0_1, N)
H2 = suite_H(*p0_2, N)


f3 = plt.figure(figsize=(10, 7))
axe3_1 = f3.add_subplot(1, 2, 1, aspect='equal')
axe3_2 = f3.add_subplot(1, 2, 2, aspect='equal')
axe3_1.set_title(f"La suite avec le terme initial {p0_1}")
axe3_2.set_title(f"La suite avec le terme initial {p0_2}")

axe3_1.plot([p[0] for p in H1], [p[1] for p in H1], ls=' ', marker='o', ms=1)
axe3_1.plot(p0_1[0], p0_1[1], color='r', ls=' ', marker='o',
            ms=3, label='terme initial')
axe3_1.plot(point[0], point[1], color='orange', ls=' ', marker='o',
            ms=3, label=f'point {point}')
axe3_1.legend()


axe3_2.plot([p[0] for p in H2], [p[1] for p in H2], ls=' ', marker='o', ms=1)
axe3_2.plot(p0_2[0], p0_2[1], color='r', ls=' ', marker='o', ms=3)


print("\nOn remarque que les termes de la suite decrivent des ensembles",
      "\nsimilaires, independamment du terme initial.\n")
```

```
210
    distance = 10**(-2)
212 n, H = premier_terme_proche(p0_1, point, distance)
    print(f"Premier terme proche de {point} pour p0 = {p0_1} est",
214         f"\n{H} d'indice {n}.\n")

216 p0 = terme_suivant_H(*H)
    n, H = premier_terme_proche(p0, point, distance)
218 print(f"Deuxieme terme proche de {point} pour p0 = {p0_1} est",
          f"\n{H} d'indice {n}.\n")
220
    n, H = premier_terme_proche(p0_2, point, distance)
222 print(f"\nPremier terme proche de {point} pour p0 = {p0_2} est",
          f"\n{H} d'indice {n}.\n")
224
    plt.show()
```