1

Enseignant·e·s: L. Testa
Informatique et Calcul Scientifique - CMS
05 juillet 2024
Durée : 150 minutes

Abra Kadabra

SCIPER : **987654**

Attendez le début de l'épreuve avant de tourner la page. Ce document est imprimé recto-verso, il contient 13 questions sur 16 pages, les dernières pouvant être vides. L'examen est sur 50 points. Ne pas dégrafer.

- Posez votre **votre pièce d'identité** sur la table.
- L'utilisation d'une **calculatrice** et de tout **outil électronique** est **interdite** pendant l'épreuve.
- Pour les questions à **choix unique**, on comptera :
  - les points indiqués si la réponse est correcte,
  - 0 point s'il n'y a aucune ou plus d'une réponse inscrite,
  - 0 point si la réponse est incorrecte.
- Vous n'avez pas besoin de commenter votre code mais vous pouvez le faire si vous pensez que cela aide à sa compréhension.
- Si une question est erronée, les enseignant·e·s se réservent le droit de l'annuler.
- Utilisez un **stylo** à encre **noire ou bleu foncé** et effacez proprement avec du **correcteur blanc** si nécessaire.
- Répondez dans l'espace prévu (**aucune** feuille supplémentaire ne sera fournie).
- Les brouillons sont à rendre, mais ils ne seront pas corrigés.

Respectez les consignes suivantes | Observe this guidelines | Beachten Sie bitte die unten stehenden Richtlinien

choisir une réponse | select an answer Antwort auswählen	ne PAS choisir une réponse | NOT select an answer NICHT Antwort auswählen	Corriger une réponse | Correct an answer Antwort korrigieren

ce qu'il ne faut **PAS** faire | what should **NOT** be done | was man **NICHT** tun sollte

Première partie, questions à choix unique

Pour chaque énoncé proposé, une ou plusieurs questions sont posées. Pour chaque question, marquez la case correspondante à la réponse correcte sans faire de ratures. Il n'y a qu'**une seule** réponse correcte par question.

Question 1 (2 points)

Qu'affiche le code ci-dessous ?

```
L1 = [4, 3, 2, 1]
L2 = [10*i for i in L1]
L3 = L2 + 2*L1

print(L3)
```

- ☐ [40, 30, 20, 10, 4, 3, 2, 1, 4, 3, 2, 1]
- ☐ [48, 36, 24, 12]
- ☐ [10, 20, 30, 40, 8, 6, 4, 2]
- ☐ [10, 20, 30, 40, 4, 3, 2, 1, 4, 3, 2, 1]
- ☐ [18, 26, 34, 42]
- ☐ [40, 30, 20, 10, 8, 6, 4, 2]

Question 2 (2 points)

Qu'affiche le code ci-dessous ?

```
def casse_tete(a, b, mid = ' ', fin = '00', *args):
    out = a + mid + b + mid + fin
    for i in args:
        out += i
    print(out, sep = '-')

casse_tete('A', 'B', '1', '2')
```

- ☐ A1B12
- ☐ A B 0012
- ☐ A B 00 1 2
- ☐ A-B-1-2
- ☐ AB0012
- ☐ A-B-00-1-2
- ☐ A-1-B-1-2
- ☐ A 1 B 1 2

Question 3 (2 points)

Qu'affiche le code ci-dessous ?

```
def chiffre(nb):
    if nb == 1:
        return "Un"
    elif nb == 2:
        return "Deux"
    elif nb > 3:
        return "Beaucoup"

print(chiffre(3))
```

- ☐ Beaucoup
- ☐ Un
- ☐ Rien ne s'affiche
- ☐ None
- ☐ Deux
- ☐ Une erreur `TypeError`

Question 4 (2 points)

On considère les deux fonctions suivantes, définies sur $\mathbb{R}$:

$$f(n) = 0.1n^2 - 10n \quad \text{et} \quad g(n) = n(1 + 3\sqrt{2n}).$$

Laquelle des affirmations suivantes est vraie?

- ☐ $g(n) = \mathcal{O}(f(n))$ mais $f(n)$ n'est pas $\mathcal{O}(g(n))$
- ☐ $f(n) = \mathcal{O}(g(n))$ mais $g(n)$ n'est pas $\mathcal{O}(f(n))$
- ☐ On peut peut pas comparer les ordres de croissance de f et de g
- ☐ $f(n) = \Theta(g(n))$

Question 5 (2 points)

Qu'affiche le code suivant?

```
import numpy as np
L1 = np.arange(6)
L2 = np.reshape(L1,(3,2))
L2 *= 2
print(L2)
```

- ☐ `[[0 2 4]`
  `[6 8 10]]`
- ☐ `[[0 2]`
  `[4 6]`
  `[8 10]]`
- ☐ `[[0 1 0 1]`
  `[2 3 2 3]`
  `[4 5 4 5]]`
- ☐ `[[0 1 2 0 1 2]`
  `[3 4 5 3 4 5]]`

Question 6 (3 points)

On donne l'algorithme ci-dessous :

```
def my_algo(L, x):
    bas = 0
    haut = len(L)-1

    while haut >= bas:
        milieu = (bas+haut)//2
        print(milieu, end = " ")
        if L[milieu] ==  x:
            return None
        if L[milieu] > x:
            haut = milieu - 1
        else:
            bas = milieu + 1
```

On définit la liste `L = [-8, -6, -4, -2, 0, 2, 4, 6, 8]`.

Quelle paire d'appels produit le même affichage?

- ☐ `my_algo(L, -1)` et `my_algo(L, -2)`
- ☐ `my_algo(L, 4)` et `my_algo(L, 5)`
- ☐ `my_algo(L, -3)` et `my_algo(L, -4)`
- ☐ `my_algo(L, 3)` et `my_algo(L, 4)`

Question 7 (2 points)

On cherche à calculer l'intégrale de la fonction $f(x) = (x-1)(x^2-3x+2)$ entre $a = -2$ et $b = 4$. Quelle méthode d'intégration numérique nous permet d'obtenir le résultat exact, c'est-à-dire avec $e_{\text{abs}} = 0$?

- ☐ La méthode des trapèzes
- ☐ La méthode du point milieu
- ☐ Aucune des trois méthodes citées
- ☐ La méthode de Simpson

Question 8 (3 points)

Parmi les cinq fonctions Python `Newton` définies ci-dessous, laquelle implémente la méthode dite de Newton permettant de déterminer une approximation d'un zéro d'une fonction `f` ?

☐
```
def Newton(f, x_0, fprime, erreur, nmax):
    x = x_0
    for n in range(0, n_max):
        if abs(f(x)) < erreur:
            return x, n+1, True
        x = x + f(x)/fprime(x)
    return x, n+1, False
```

☐
```
def Newton(f, x_0, fprime, erreur, nmax):
    x = x_0
    for n in range(0, n_max):
        if abs(f(x)) < erreur:
            return x, n+1, True
        x = x - f(x)*fprime(x)
    return x, n+1, False
```

☐
```
def Newton(f, x_0, fprime, erreur, nmax):
    x = x_0
    for n in range(0, n_max):
        if abs(f(x)) < erreur:
            return x, n+1, True
        x = x + fprime(x)/f(x)
    return x, n+1, False
```

☐
```
def Newton(f, x_0, fprime, erreur, nmax):
    x = x_0
    for n in range(0, n_max):
        if abs(f(x)) < erreur:
            return x, n+1, True
        x = x - f(x)/fprime(x)
    return x, n+1, False
```

☐
```
def Newton(f, x_0, fprime, erreur, nmax):
    x = x_0
    for n in range(0, n_max):
        if abs(f(x)) < erreur:
            return x, n+1, True
        x = x - fprime(x)/f(x)
    return x, n+1, False
```

Question 9 (2 points)

Qu'affiche le code ci-dessous ?

```
L1 = [['A'], ['B']]
L2 = L1.copy()
L2.append(['D'])
L1[1].append('C')
print (L1, L2,sep='\n')
```

☐
```
[['A'], ['B'], ['C']]
[['A'], ['B'], ['C'], ['D']]
```

☐
```
[['A'], ['B', 'C']]
[['A'], ['B', 'C'], ['D']]
```

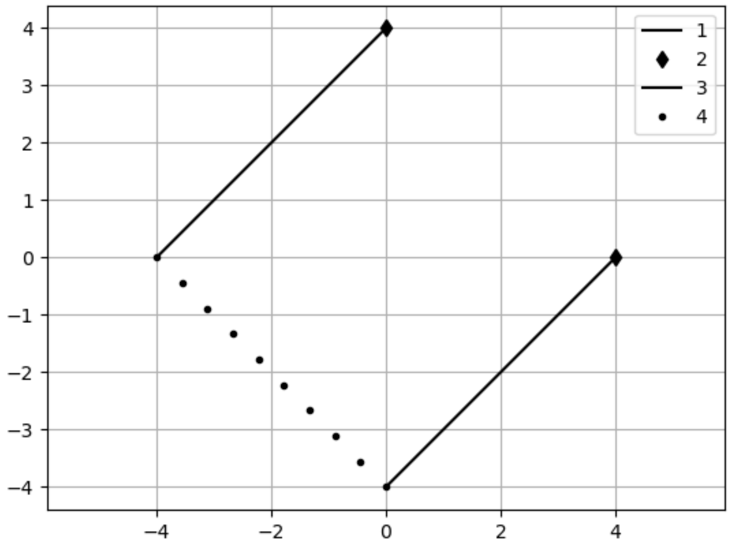
☐
```
[['A'], ['B', 'C'], ['D']]
[['A'], ['B', 'C'], ['D']]
```

☐
```
[['A'], ['B'], ['C'], ['D']]
[['A'], ['B'], ['C'], ['D']]
```

Question 10 (3 points)

Parmi les huit fonctions Python `my_plot` proposées, laquelle produit la figure ci-dessous lorsqu’elle est appelée dans le code suivant ?

```
plt.figure()
###
my_plot()
###
plt.grid()
plt.axis("equal")
plt.show()
```



☐
```
def my_plot():
    plt.plot([0,4],[-4,0] ,'k',label='1')
    plt.plot([0,-4],[4,0],'k-',label='3')
    plt.plot([-4,0],[0,-4],'k--',label='4')
```

☐
```
def my_plot():
    plt.plot([0,4],[-4,0] ,'k',label='1')
    plt.plot([4,0],[0,4],'kd',label='2')
    plt.plot([0,-4],[4,0],'k-',label='3')
    plt.plot(x,-x-4,'k.',label='4')
    plt.legend()
```

☐
```
def my_plot():
    plt.plot([0,4],[-4,0] ,'k',label='1')
    plt.plot([0,-4],[4,0],'k-',label='3')
    plt.plot(x,-x-4,'k.',label='4')
```

☐
```
def my_plot():
    plt.plot([0,4],[-4,0] ,'k',label='1')
    plt.plot([4,0],[0,4],'kd',label='2')
    plt.plot([0,-4],[4,0],'k-',label='3')
    plt.plot([-4,0],[0,-4],'k--',label='4')
    plt.legend()
```

☐
```
def my_plot():
    plt.plot([0,4],[-4,0] ,'k',label='1')
    plt.plot([4,0],[0,4],'kd',label='2')
    plt.plot([0,-4],[4,0],'k-',label='3')
    plt.plot(x,-x-4,'k.',label='4')
```

☐
```
def my_plot():
    plt.plot([0,4],[-4,0] ,'k',label='1')
    plt.plot([4,0],[0,4],'kd',label='2')
    plt.plot([0,-4],[4,0],'k-',label='3')
    plt.plot([-4,0],[0,-4],'k--',label='4')
```

☐
```
def my_plot():
    plt.plot([0,4],[-4,0] ,'k',label='1')
    plt.plot([0,-4],[4,0],'k-',label='3')
    plt.plot([-4,0],[0,-4],'k--',label='4')
    plt.legend()
```

☐
```
def my_plot():
    plt.plot([0,4],[-4,0] ,'k',label='1')
    plt.plot([0,-4],[4,0],'k-',label='3')
    plt.plot(x,-x-4,'k.',label='4')
    plt.legend()
```

Deuxième partie, questions de type ouvert

Répondez dans l'espace dédié. Laissez libres les cases à cocher : elles sont réservées à la correction.

Question 11: *Cette question est notée sur 6 points.*

☐.5 ☐.5 ☐.5 ☐.5 ☐.5 ☐.5
☐0 ☐1 ☐2 ☐3 ☐4 ☐5 ☐6

On considère une liste `L` ne contenant que des éléments de type `str`. Par exemple,
`L = ["Ceci", "est", "un", "exemple", "de", "liste"]`.

On aimerait classer ces éléments en fonction de leur longueur dans un dictionnaire au travers de plusieurs étapes.

(a) Ecrivez un programme permettant d'extraire la longueur `lmax` de la chaîne de charactères la plus longue de cette liste.

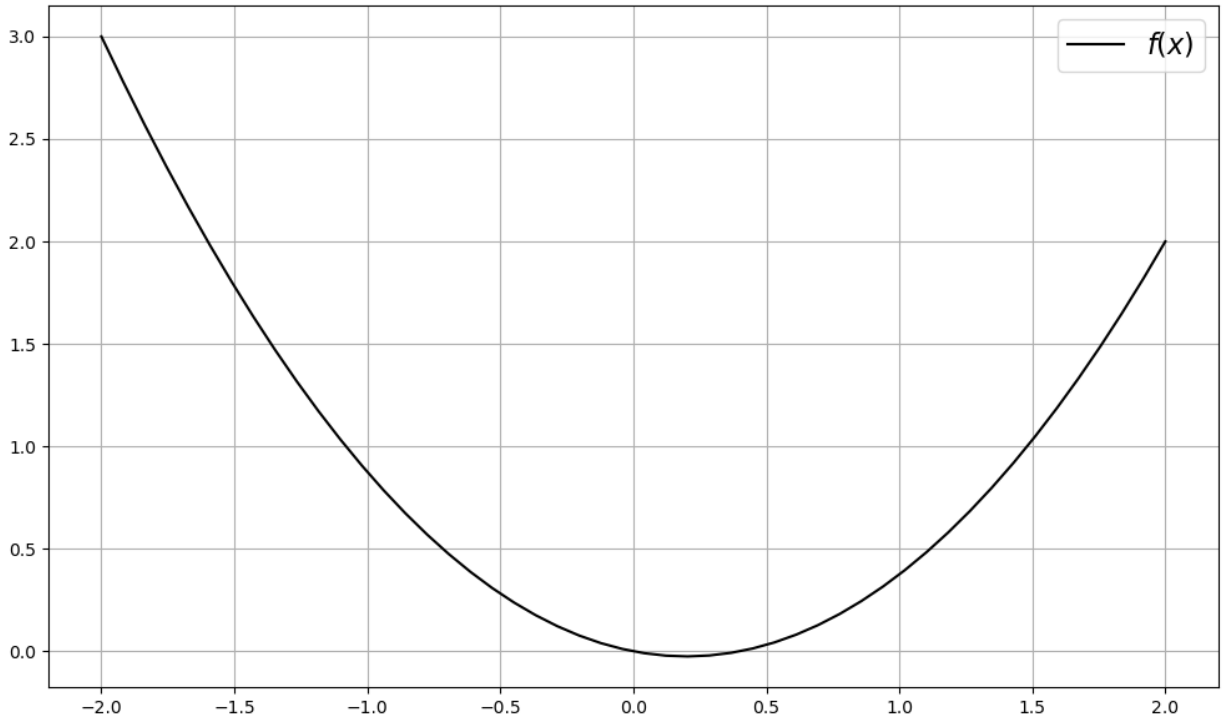
(b) Initialisez un dictionnaire `d` dont les différentes clés sont des entiers allant de `1` à `lmax` et les valeurs associées sont des listes vides.

(c) Remplissez le dictionnaire `d` selon la logique suivante : la valeur associée à la clé `N` doit contenir tous les mots de `L` contenant `N` lettres. En reprenant l'exemple précédent,
`d = {1: [], 2: ['un', 'de'], 3: ['est'], 4: ['Ceci'], 5: ['liste'], 6: [], 7: ['exemple']}`.

Question 12: *Cette question est notée sur 11 points.*

.5 .5 .5 .5 .5 .5 .5 .5 .5 .5 .5
0 1 2 3 4 5 6 7 8 9 10 11

On considère la fonction $f(x)$ représentée graphiquement ci-dessous sur l'intervalle $[-2, 2]$.



(a) En utilisant la base de Lagrange appropriée, trouvez l'expression analytique de $f(x)$, en sachant qu'il s'agit d'un polynôme de degré 2 qui en $x_0 = -2$ vaut 3, en $x_1 = 0$ vaut 0 et en $x_2 = 2$ vaut 2.

(b) Calculez "à la main", c'est-à-dire sans écrire de code, l'intégrale de $f(x)$ entre -2 et 2 en utilisant la méthode de quadrature composite basée sur la formule des trapèzes et en considérant une partition régulière du domaine d'intégration $[-2, 2]$ en deux sous-intervalles. Le résultat doit être sous forme de valeur numérique.

(c) Complétez le code Python ci-dessous définissant une fonction Python `integration_Simpson` permettant d'approcher l'intégrale d'une fonction `f` sur le domaine `[a,b]` en utilisant la méthode de quadrature composite avec `n` sous-intervalles basée sur la formule de Simpson.

```
# CALCUL NUMERIQUE DE L'INTEGRALE DEFINIE
# EN UTILISANT LA METHODE DE SIMPSON

def integration_Simpson(f,a,b,n):
    '''
    PARAMETRES
    f :             Fonction a integrer
    a, b :          Bornes du domaine d'integration
    n :             Nombre de sous-intervalles
    VARIABLES
    I :             Approximation de l'integrale
    dx :            Finesse de la partition
    xmin, xmax :    bornes du sous-intervalle
    '''
    # 1. INITIALISATION DES VARIABLES

    I = _______________

    dx = _______________

    xmax = _______________

    # 2. CALCUL DES INTEGRALES
    for i in range(1,n+1):

        xmin = _______________

        xmax = _______________

        I += _______________

    return _______________
```

(d) Est-il possible de déterminer la valeur exacte de l'erreur absolue obtenue $e_{\mathbf{abs}}$ en appliquant la méthode de Simpson à la fonction `f(x)` définie à la page précédente, et en considérant une partition régulière du domaine d'intégration $[-2, 2]$ en trois sous-intervalles? Si oui, que vaut-elle?

Question 13: *Cette question est notée sur 10 points.*

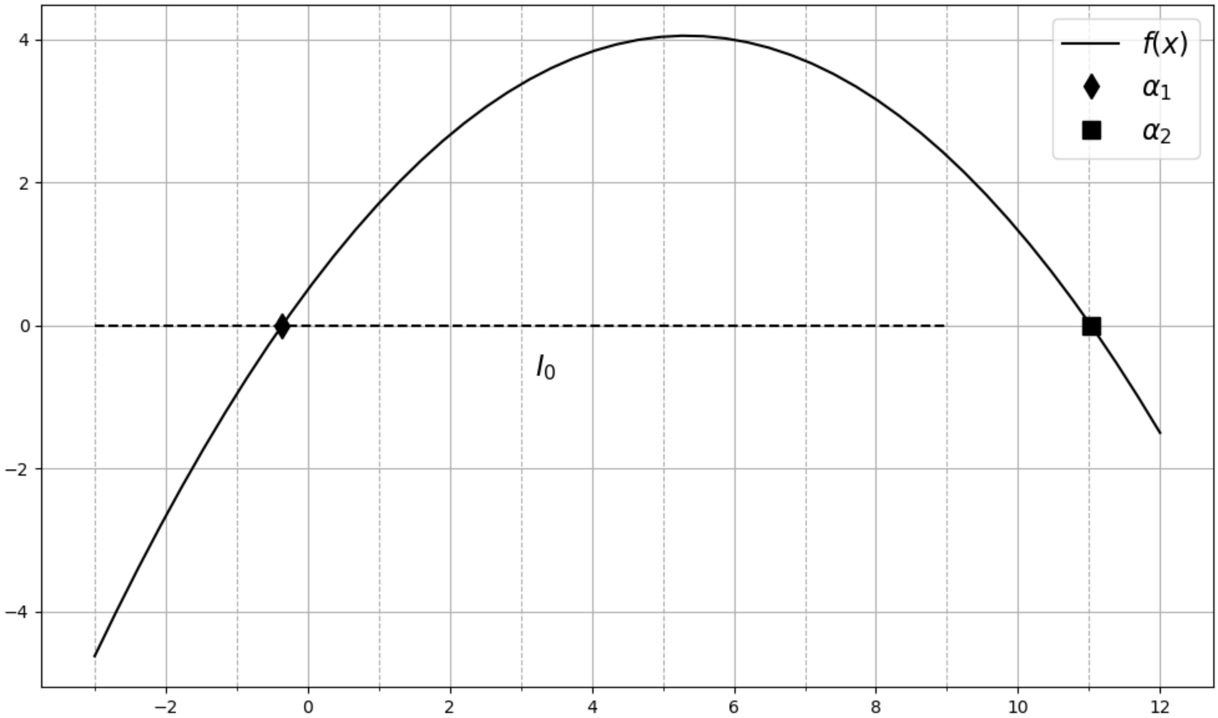
.5 .5 .5 .5 .5 .5 .5 .5 .5 .5
0 1 2 3 4 5 6 7 8 9 10

On considère la fonction $f : \mathbb{R} \to \mathbb{R}$ à variable réelle définie par

$$f(x) = \frac{4}{3}x + \frac{2 - \frac{1}{2}x^2}{4},$$

et représentée graphiquement ci-dessous sur l'intervalle $[-3, 13]$.

On aimerait obtenir une approximation des racines α_1 et α_2 à l'aide de la méthode numérique de la bissection, en partant de l'intervalle $I_0 = [a_0, b_0] = [-3, 9]$ représenté sur la même image.



(a) En partant de I_0, définissez l'intervalle considéré lors des quatre itérations suivantes de la méthode, c'est-à-dire $I_k = [a_k, b_k]$ avec $k = 1, 2, 3, 4$.

(b) Que vaut x_3, l'approximation de la racine obtenue si on termine l'algorithme après la troisième itération de la méthode de la bissection, c'est-à-dire après $k = 3$.
Que vaut l'erreur absolue maximale $e_{\text{abs},3}$ correspondante?

(c) Quel est le nombre minimal d'itérations à effectuer pour que l'erreur sur l'approximation de la racine satisfasse une tolérance de $\varepsilon = \frac{\mathbf{3}}{\mathbf{16}}$, donc soit telle que $e_{\text{abs},\boldsymbol{k}} = \varepsilon$?

(d) On considère maintenant un nouvel intervalle de départ, $\boldsymbol{I_0} = [\mathbf{-1, 12}]$. Est-ce que la méthode de la bissection simple peut nous permettre de trouver une bonne approximation de $\boldsymbol{\alpha_2}$? Justifiez mathématiquement.

(e) Afin de trouver une valeur approchant α_2, pourrait-on utiliser la méthode de Newton en partant de $x_0 = \frac{16}{3}$? Justifiez mathématiquement.