+1/1/60+

EPFL

1

**Enseignant: L. Testa**
**Informatique et Calcul Scientifique (ICS) - MAN**
**Printemps 2024**
**Durée : –**

# Dalton Joe

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. Ne pas dégrafer. L'examen comporte un total de XX points.**

- Posez votre **carte d'étudiant.e** 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 si il n'y a aucune ou plus d'une réponse inscrite,
  0 point si la réponse est incorrecte.
- Les algorithmes demandés sont à écrire sous forme de fonctions Python. Mettez votre code en forme en respectant les indentations: 1 carreau = 1 espace (donc 4 carreaux = 1 tab).
- Vous n'avez pas besoin de commenter votre code mais vous pouvez le faire si vous pensez que cela aide à sa compréhension.
- 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 ne sont pas à rendre: 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<br>Antwort auswählen | ne PAS choisir une réponse \| NOT select an answer<br>NICHT Antwort auswählen | Corriger une réponse \| Correct an answer<br>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, marquer la case correspondante à la réponse correcte sans faire de ratures. Il n'y a qu'**une seule** réponse correcte par question.

Cette première partie évalue vos connaissances sur le langage Python.
Qu'affiche chacun des codes ci-dessous?

**Question 1** (2 points)

```
try:
    new = []
    new.append(1+"2")
    a = new[1]/0
    a = 1
    print(a)
except ZeroDivisionError :
    print("2")
except IndexError:
    print("3")
except TypeError:
    print("4")
```

- ☐ 4
- ☐ 2
- ☐ 1
- ☐ 3

**Question 2** (2 points)

```
def mystere(*args):
    d = {}
    j = 0
    for i in range(len(args)-1, -1, -1):
        d[j] = args[i]
        j += 1
    return d

print(mystere(3, 5, 7, 9, 11))
```

- ☐ {11:0, 9:1, 7:2, 5:3, 3:4}
- ☐ {11:3, 9:5, 7:7, 5:9, 3:11}
- ☐ {0:11, 1:9, 2:7, 3:5, 4:3}
- ☐ {3:3, 5:5, 7:7, 9:9, 11:11}

**Question 3** (2 points)

```
L1 = [0, 1, 2]
L2 = [5, 5, 5]

L3 = [2*x+y for x in L1 for y in L2]

print(L3)
```

- ☐ `[9, 11, 13, 9, 11, 13, 9, 11, 13]`
- ☐ `[5, 7, 9, 5, 7, 9, 5, 7, 9]`
- ☐ `[5, 7, 9]`
- ☐ `[5, 5, 5, 7, 7, 7, 9, 9, 9]`
- ☐ `[9, 11, 13]`
- ☐ `[9, 9, 9, 11, 11, 11, 13, 13, 13]`

**Question 4** (2 points)

```
i = 2
while i < 10:
    if i%2 == 0:
        print(i,end=' ')
        if i == 6:
            break
```

- ☐ 2 4 6 8
- ☐ 2 3 4 5 6 7 8 9
- ☐ 2 4 6
- ☐ Aucune de ces réponses

**Question 5** (2 points)

```
def affichages(d):
    for x in d:
        print(d[x], end = " ")
    print()
    for x in d.items():
        print(x[1], end = " ")
    print()
    for x in d.values():
        for y in d:
            if d[y] == x:
                print(y, end = " ")

d1 = {"a":10, "b":20, "c":30}
affichages(d1)
```

- ☐ a b c
  a b c
  a b c
- ☐ a b c
  10 20 30
  a b c
- ☐ 10 20 30
  10 20 30
  a b c
- ☐ a b c
  a b c
  10 20 30
- ☐ 10 20 30
  10 20 30
  10 20 30
- ☐ 10 20 30
  a b c
  10 20 30

**Question 6** (2 points)

```
def surprise(t):
    s = 0
    for x in t:
        s += x[1]
    return s

t1 = ((1, 2), (3, 4), (5, 6), (7, 8))
print(surprise(t1))
```

- ☐ 36
- ☐ 20
- ☐ 4
- ☐ 16

**Question 7** (2 point)

```
L = ["a", "b", "a", "a", "b"]
for i in range(1, 4):
    L[i] += L[i+1]
for s in ("ab", "ba"):
    if s in L:
        print(s, L.count(s))
```

- ☐ ab 1<br>ba 1
- ☐ ab 2<br>ba 1
- ☐ ab 2<br>ba 2
- ☐ L'affichage produit une exception `IndexError`

**Question 8** (2 points)

```
for i in range(4):
    for j in range(i):
        print('*',sep='|', end = '.')
    print()
```

- ☐
```
*.*.*.
*.*.
*.
```
- ☐
```
*|*|*|.
*|*|.
*|.
```
- ☐
```
*|.
*|*|.
*|*|*|.
```
- ☐
```
*.
*.*.
*.*.*.
```

**Question 9** (2 points)

```
L = [[0], [0]]
L_prime = L.copy()
L.append([1])
L[1].append(1)
print(L, L_prime)
```

- ☐ `[[0], [0, 1], [1]] [[0], [0, 1]]`
- ☐ `[[0], [0, 1], [1]] [[0], [0, 1], [1]]`
- ☐ `[[0], [0, 1], [1]] [[0], [0], [1]]`
- ☐ `[[0], [0, 1], [1]] [[0], [0]]`

**Question 10** (2 points)

```
def oui_non(L, L1 = [1, 3]):
    for x in L:
        for y in L1:
            if x < y:
                print("oui", end = " ")
            else:
                print("non", end = " ")

oui_non([2, 4])
```

- ☐ non oui non non
- ☐ oui non oui oui
- ☐ non non
- ☐ oui oui

**Question 11** (2 points)

On considère la modification suivante de l'algorithme de recherche binaire vu en cours.

```
def recherche_binaire(L ,x):

    n = len(L)
    bas = 0
    haut = n-1
    count = 0
    while haut >= bas :
        count += 1
        milieu = (bas+haut)//2
        if L[milieu] == x:
            return count
        elif L[milieu] > x:
            haut = milieu - 1
        else :
            bas = milieu + 1
    return float('inf')
```

Soit la liste `L = [-8, -5, -4, -2, 0, 1, 2, 9, 12, 15, 18, 20, 26, 28, 32, 35]`.
Qu'affiche l'instruction `print(recherche_binaire(L,1))`?

- ☐ `0`
- ☐ `4`
- ☐ `1`
- ☐ `2`
- ☐ `3`
- ☐ `'inf'`

**Question 12** (2 points)
Qu'affiche le code suivant ?

```
def my_func(L):

    n = len(L)
    bas = 0
    haut = n-1
    while haut >= bas :
        milieu = (bas+haut)//2
        if L[milieu] == milieu:
            bas = milieu + 1
        else :
            haut = milieu - 1
    return bas

L = [0, 1, 2, 3, 4, 5]
print(my_func(L))
```

- ☐ `0`
- ☐ `2`
- ☐ `6`
- ☐ `4`

**Question 13** (2 points)

On considère les deux fonctions suivantes définies sur les entiers strictements supérieurs à 1:

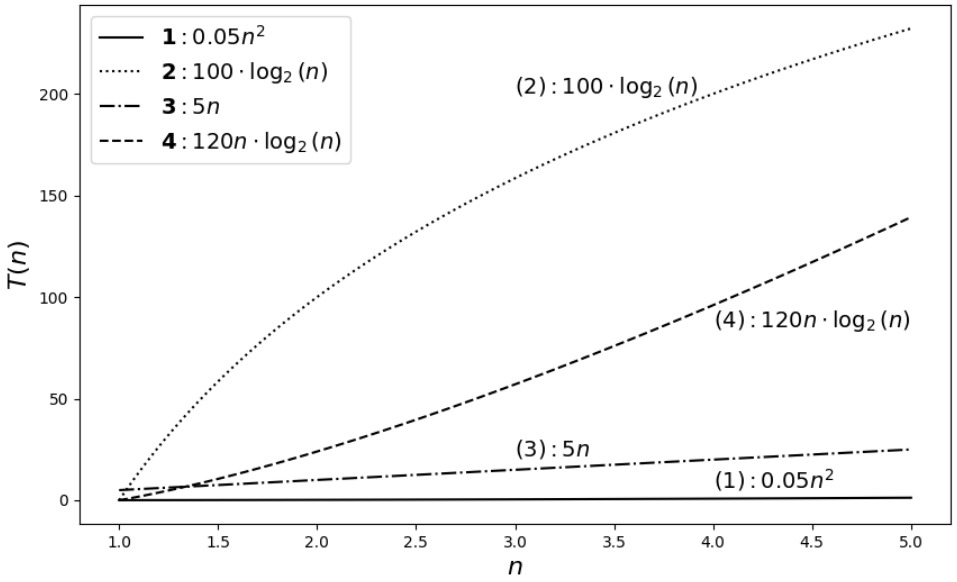
$$f(n) = \frac{n}{\log_2(n)} \text{ et } g(n) = \sqrt{n}\log_2(n).$$

Laquelle des affirmations suivantes est vraie?

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

**Question 14** (2 points)

Le graphe suivant représente les temps de parcours de quatre algorithmes différents pour résoudre le même problème.




Ordonnez les quatre algorithmes du plus efficace au moins efficace.

- ☐ 1 3 4 2
- ☐ 4 2 3 1
- ☐ 2 3 1 4
- ☐ 2 4 3 1
- ☐ 2 3 4 1
- ☐ 1 3 2 4

## Deuxième partie, questions de type ouvert

Répondre dans l'espace dédié. Laisser libres les cases à cocher : elles sont réservées au correcteur.

**Question 15:** *Cette question est notée sur 8 points.*

Dans cet exercice, on va écrire un entier strictement positif quelconque $N$ sous la forme $N = 2^p + r$, avec $p$ le plus grand entier possible et $0 \leq r < 2^p$. Pour cela, on procède en plusieurs étapes.

**Note** : Cet exercice comprend deux sous-questions, chacune peut être résolue indépendamment.

(a) Ecrivez une fonction `decomposition` qui prend comme paramètre un nombre entier strictement positif $N$ (pas besoin de tester cette condition dans le corps de la fonction), et qui divise ce nombre par deux tant que le résultat de cette opération n'est pas strictement inférieur à 1. Cette fonction doit retourner

(i) le nombre $p$ de divisions effectuées

(ii) le reste $r$ de cette opération tel que $N = 2^p + r$.

Par exemple, l'instruction `decomposition(4)` doit retourner `(2,0)` car `4 = 2*2+0`, `decomposition(7)` doit retourner `(2,3)` car `7 = 2*2+3` et `decomposition(9)` doit retourner `(3,1)` car `9 = 2*2*2+1`.

(b) Ecrivez un programme qui

(i) génère un nombre aléatoire compris entre `1` et `100`

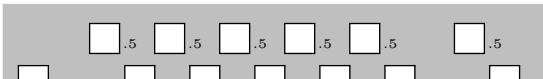
(ii) appelle la fonction définie au point (a)

(iii) affiche le résultat à l'écran sous la forme : `N = 2**p+r`.

Par exemple, `16 = 2**4+0` et `15 = 2**3+7`.

**Indication:** On rappelle que la fonction `randint(a,b)` du module `random` de la bibliothèque standard de Python prend comme paramètres deux nombres `a` et `b` et retourne un nombre aléatoire appartenant à l'intervalle `(a,b)`.

**Question 16:** *Cette question est notée sur 6 points.*



Ecrivez une fonction `list_to_dict` qui prend en entrée une liste `L` de nombres, et retourne un dictionnaire `d` tel que:

- Les clés de `d` sont les éléments de `L`
- La valeur associée à une clé `x` est **la liste des indices** où `x` apparaît dans la liste `L`.

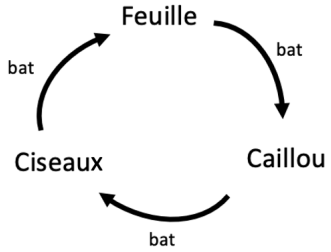
Par exemple,

- Pour la liste `[10, 20, 10, 30]` en entrée, `list_to_dict` doit retourner `{10:[0, 2], 20:[1], 30:[3]}`
- Pour une liste vide en entrée, `list_to_dict` doit retourner un dictionnaire vide.

Il n'y a pas besoin de vérifier que l'argument fourni en entrée à la fonction `list_to_dict` est bien une liste de nombres.

**Question 17:** *Cette question est notée sur 13 points.*

Rappelons les règles du Feuille Caillou Ciseaux, aussi appelé "Rock Paper Scissors" ou Shifumi. Chaque joueur choisit un coup parmi les trois possibles: Feuille, Caillou ou Ciseaux, en sachant que la feuille bat le caillou, le caillou bat les ciseaux et les ciseaux battent la feuille, comme indiqué dans le diagramme ci-dessous:




Si on représente Feuille par 0, Caillou par 1 et Ciseaux par 2, le tableau suivant présente les différents coups possibles ainsi que le coup gagnant pour chaque possibilité.

| | 0 | 1 | 2 |
|---|---|---|---|
| 0 | Egalité | 0 | 2 |
| 1 | 0 | Egalité | 1 |
| 2 | 2 | 1 | Egalité |

Vous souhaitez écrire un programme pour permettre à votre amie Alice de jouer à Feuille Caillou Ciseaux à distance avec Bob qui se situe à l'autre bout du monde. Pour cela, Bob vous envoie par email un fichier `fcc.txt` contenant ses 5 prochaines actions, que vous placez dans le dossier courant, c'est-à-dire le dossier où vous exécuterez votre fichier .py ou votre Jupyter Notebook. De plus, vous convenez ensemble du code Feuille = 0, Caillou = 1 et Ciseaux = 2.

Par exemple si le contenu de fichier est comme ci-dessous, cela indique que Bob jouera Feuille au premier tour, Caillou aux deux suivants, puis Feuille et finalement Ciseaux.

**Fichier 'fcc.txt'**

```
0
1
1
0
2
```

**Note** : Cet exercice comprend cinq sous-questions, chacune peut être résolue indépendamment.

(a) Commencez par importer les coups de Bob, et stockez-les dans la liste `ami`.

(b) Créez une fonction `testinput()` qui demande à l'utilisatrice (Alice) d'entrer un nombre entier compris entre 0 et 2 inclus. Tant que la valeur entrée ne satisfait pas ces conditions, votre fonction doit continuer à demander un nombre à Alice. Elle doit finalement retourner cette valeur.
Si Alice rentre `4`, `2.1` ou `"deux"`, le programme doit afficher : `On avait dit 0, 1 ou 2 !` et lui redemander d'entrer un nombre.

(c) Ecrivez un programme qui, pour chaque coup joué par Bob, demande à Alice d'entrer un nombre compris entre 0 et 2, compare ce nombre avec celui joué par Bob, et stocke le nom du vainqueur (Alice, Bob ou Egalité) dans une liste `resultat`. Si Alice a joué `[0, 0, 0, 0, 0]`, alors `resultat` doit contenir `["Egalite", "Alice", "Alice", "Egalite", "Bob"]`.
Vous pouvez utiliser directement la variable `ami` ainsi que la fonction `testinput()`.

(d) Une fois les cinq tours effectués, écrire les résultats dans un fichier `final.txt`. Chaque ligne de ce fichier doit correspondre au résultat d'un tour.

(e) A la fin de la partie, Bob vous renvoie un nouveau fichier avec ses 5 prochaines actions. Que devons-nous modifier au code du point (d) pour rajouter le résultat de ces 5 parties à la fin du fichier `final.txt`?

**Question 18:** *Cette question est notée sur 7 points.*

On donne l'algorithme ci-dessous.

```
def mafonction(L):
    n = len(L)
    for i in range(n-1,-1,-1):
        j = i
        while j < n-1 and L[j] > L[j+1]:
            L[j], L[j+1] = L[j+1], L[j]
            j += 1

        print(f"{i}e element: {L}")
```

(a) Qu'affichent les instructions suivantes?

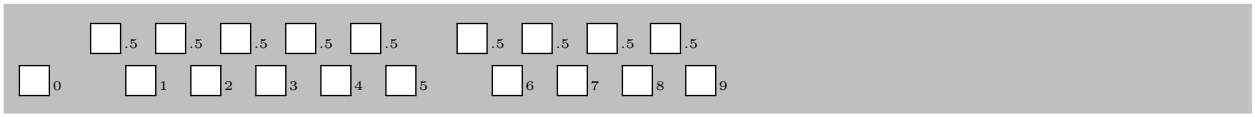
```
L = [1, 3, 0, -6, 100, -1]
mafonction(L)
```

(b) Quel algorithme vu en cours fonctionne selon le même principe que celui-ci?

(c) On dénote par $T(n)$ le temps de parcours de l'algorithme **`mafonction()`** lorsqu'il prend en entrée une liste de taille $n$, dans le pire des cas. Donner l'ordre de croissance de $T(n)$ en notation $\Theta(\cdot)$, en justifiant brièvement votre réponse.

(d) Pour quelles instances le temps de parcours est-il minimal? Donner l'ordre de croissance du temps de parcours **dans le meilleur des cas** en notation $\Theta(\cdot)$, en justifiant brièvement votre réponse.

**Question 19:** *Cette question est notée sur 9 points.*



La fonction **surprise()** ci-dessous est une implémentation d'un algorithme itératif qui prend un nombre réel **N** en entrée.

```
def surprise(N):
    co = 0
    N /= 2
    while N>=1:
        co += 1
        N /= 2
    return co
```

(a) Qu'affichent les instructions suivantes?

```
L = [1, 1.7, 2, 5, 8, 15]

for i in L:
    print(surprise(i))
```

(b) Soit $T(N)$ le temps de parcours de la fonction **`surprise()`** en fonction de **la valeur** du nombre en entrée $N$. Donnez l'ordre de croissance de $T(N)$ en notation $\Theta(\cdot)$, en justifiant brièvement votre réponse.