EPFL

1

Ens. : O. Lévêque, M. Stojilovic
Information, Calcul, Communication - GC
Vendredi 20 décembre 2024
Durée : 180 minutes

n/a

SCIPER : **999999**

Salle :

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

- Posez votre carte d'étudiant sur la table.
- Document autorisé pour cet examen : un formulaire constitué de deux pages A4 recto-verso, manuscrites, préparées avec stylet + tablette ou à l'ordinateur.
- L'utilisation de tout appareil électronique (calculatrice, ordinateur, smartphone/watch, tablette) est interdite pendant l'épreuve.
- L'examen est composé de deux parties:
  - une partie avec 13 questions à choix multiple valant en tout 32 points ; chaque question admet une seule réponse correcte : la réponse correcte vaut 2 points (pour les 7 premières questions sur la partie théorique) ou 3 points (pour les 6 questions suivantes sur la partie programmation) ; toute autre option (pas de réponse, réponse fausse, ou plusieurs cases cochées) vaut 0 point.
  - une partie avec 5 questions de type ouvert, valant en tout 48 points.
- Merci d'avance de soigner la présentation de vos réponses !
- Si une question est erronée, les enseignants se réservent le droit de l'annuler.

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

Question 8

Qu'affiche ce programme ?

```
def addme(x, y):
    global cnt
    cnt += 1

    res.add(x + y)

    if x > 0 and y > 0:
        addme(x - 2, y)
        addme(x, y - 1)

cnt = 0
res = set()
addme(4, 2)
print(cnt, len(res))
```

☐ 7 3 ☐ 11 11 ☐ 5 3 ☐ 11 6 ☐ 3 3

Question 9

Qu'affiche ce programme ?

```
def foo(a):
  b = 100
  c = a + b or a - b
  a, b, c = b, c, a
  return a, b, c

a = 10
b = a + 10
c = b + 10

x, y, z = foo(b)
print(y + z, a + b)
```

☐ 120 30 ☐ 220 110 ☐ 140 220 ☐ 40 30 ☐ 140 30 ☐ 21 30

Question 10

Qu'affiche ce programme ?

```
i = 1
new_dict = dict.fromkeys(range(6), i)
elements = list(new_dict.keys())

for elem in elements:
    i += 1
    for e in elements[::i]:
        new_dict[elem] -= e

new_dict.popitem()
new_dict[new_dict.pop(2)] = i

print(len(new_dict), set(new_dict.values()))
```

- ☐ 6 {1}
- ☐ 5 {1, 7, -5, -4, -2}
- ☐ 6 {1, 7, -5, -4, -2}
- ☐ 5 {1, -5, -4, -3, -2}
- ☐ 6 {1, 7, -5, -4, -3, -2}
- ☐ 6 {1, -5, -4, -3, -2}

Question 11

Quel pourrait être le résultat de ce programme ?

```
lst = []
for i in range(6):
    for j in range(i):
        lst.append(i)
s = set(lst)

s = (s & {4, 5, 6, 7}) - {6, 7}
s = (s | {8}) ^ {4, 9}

print(s)
```

- ☐ {1, 2, 3, 5, 8, 9}
- ☐ {8, 5, 9}
- ☐ {5, 6, 7, 8, 9}
- ☐ {9, 4}
- ☐ {5, 8, 6, 7}

Question 12

Qu'affiche ce programme ?

```
def bar(n, r = 1):
    if n == 0:
        return r
    elif n % 2:
        return bar(n - 1, r + n)
    else:
        return bar(n - 1, r + 2 * n)

print(bar(10))
```

☐ 56 ☐ 81 ☐ 79 ☐ 85 ☐ 86

Question 13

Qu'affiche ce programme ?

```
def matrix_manipulation(v_in, matrix_in, matrix_out):
    global n
    for i in range(n):
        for j in range(n):
            if i == 2:
                continue
            if j == 1:
                break
            matrix_out[j][0] += matrix_in[j][i] * v_in[i]

n = 3
vector = [1, 2, 3]
matrix_a = [[1, 2, 3], [1, 3, 2], [2, 1, 3]]
matrix_b = [[0] for i in range(3)]

matrix_manipulation(vector, matrix_a, matrix_b)
print(matrix_b)
```

☐ `[[3], [0], [0]]` ☐ `[[14], [0], [0]]` ☐ `[[1], [4], [0]]`

☐ `[[5], [0], [0]]` ☐ `[[5], [7], [4]]`

Question 17: *Cette question est notée sur 11 points.*

☐0 ☐1 ☐2 ☐3 ☐4 ☐5 ☐6 ☐7 ☐8 ☐9 ☐10 ☐11

Écrivez une fonction Python `aggregate_score()` qui prend comme argument une liste de chaînes de caractères. Chaque chaîne de la liste contient le nom d'un étudiant suivi d'un espace et d'un score représentant les points obtenus. Votre fonction doit agréger les notes correspondant à un même étudiant et renvoyer un dictionnaire contenant la note agrégée de chaque étudiant.

Données d'entrée : Une liste de chaînes de caractères, chaque chaîne étant formatée comme « nom score », où nom est le nom de l'étudiant et score est un nombre entier non négatif. Vous pouvez supposer que les noms sont composés d'un seul mot sans espace et que chaque nom commence par une majuscule. Voici un exemple de liste de notes : `["Tom 5", "Lea 10", "Tom 7", "Lea 15"]`. Étant donné que les noms commencent par une majuscule, la situation dans laquelle le dictionnaire contient, par exemple, les noms `"Tom"` et `"tom"`, ne peut pas se produire.

Résultat : Un dictionnaire dans lequel chaque clé est un nom unique de la liste d'entrée et la valeur correspondante est la somme de tous les scores associés à ce nom. Par exemple, la sortie de l'entrée donnée sera `{'Tom' : 12, 'Lea' : 25}`.

Exemple d'utilisation :

```
scores = ["Tom 5", "Lea 10", "Tom 7", "Lea 15"]
score_dict = aggregate_score(scores)
print(score_dict)
# Output: {'Tom': 12, 'Lea': 25}
```

Question 18: *Cette question est notée sur 11 points.*

☐0 ☐1 ☐2 ☐3 ☐4 ☐5 ☐6 ☐7 ☐8 ☐9 ☐10 ☐11

On vous donne deux listes, `list1` et `list2`, toutes deux de longueur n, et une fenêtre de taille w. Votre tâche consiste à écrire une fonction Python **récursive** `window_average_recursive()` pour calculer une nouvelle liste dont chaque élément représente la moyenne des valeurs des deux listes d'entrée à l'intérieur d'une fenêtre coulissante de taille w.
Étapes de la résolution :

(a) Pour chaque position i de la fenêtre coulissante ($0 \leq i \leq n - w$) :

- Considerez les sous-sections :

$$\text{window1} = \text{list1}[i], \text{list1}[i+1], \ldots, \text{list1}[i+w-1]$$

et

$$\text{window2} = \text{list2}[i], \text{list2}[i+1], \ldots, \text{list2}[i+w-1].$$

- Calculer les moyennes par élément :

$$\text{element\_average}[j] = \frac{\text{window1}[j] + \text{window2}[j]}{2}, \quad j = 0, 1, \ldots, w-1.$$

- Calculer la moyenne globale de ces valeurs à l'intérieur de la fenêtre :

$$\text{window\_average} = \frac{1}{w} \sum_{j=0}^{w-1} \text{element\_average}[j].$$

- Ajouter window_average à la liste des résultats.

(b) Répéter ce processus pour toutes les positions possibles de la fenêtre coulissante jusqu'à ce que $i = n-w$.

Résultat : La liste résultante aura une longueur de $n-w+1$. Le programme doit renvoyer une liste contenant toutes les moyennes de la fenêtre.
Exemple d'utilisation :

```
list1 = [1, 2, 3, 4, 5]
list2 = [0, 2, 1, 3, 8]
w = 3
result = window_average_recursive(list1, list2, w)
print(result)
# Output: [1.5, 2.5, 4.0]
```

Notez que, si l'implémentation de votre code est entièrement correcte mais ne se présente pas sous la forme d'une fonction récursive, le nombre maximum de points que vous pouvez obtenir pour cette question est 5.