22 mai 2025 IC1T

Examen de programmation

Durée: 2h Version: **A**

Instructions

— Répondez sur la feuille de réponse séparée **à votre nom**. **Aucune** réponse écrite sur le présent énoncé ne sera prise en compte dans la correction
— Répondez en noircissant la case de la réponse avec un **stylo ou un feutre** (pas de crayon).
— Noircissez aussi la version (**A** ou **B**) de votre énoncé sur la feuille de réponse.
— Pour corriger une réponse, **effacez complètement** la mauvaise case avec un typex avant d'en noircir une autre.
— Vous ne devez pas remettre l'énoncé. Vous pouvez donc l'utiliser comme brouillon.
— Il n'y a pas de points négatifs.
— **Aucun** document n'est autorisé.

Question 1 (1 point)

Laquelle des propositions suivantes est fausse ?

A) En Python, on peut multiplier une liste par un entier

B) Un programme est une suite d'instructions visant à la résolution d'un problème.

C) La mémoire RAM est volatile.

D) Une variable globale est définie en dehors de toute fonction

E) Un SSD est un stockage de masse

F) Un pseudo-code n'est écrit dans aucun langage particulier.

G) `Desktop\main.py` est un chemin relatif

H) Le type `complex` existe en Python

I) Le stockage de masse est généralement plus petit que la mémoire

J) La commande `pwd` permet d'afficher le répertoire courant

Question 2 (1 point)

Que vaut `y` à la fin de l'exécution du code suivant ?

```
1  import numpy as np
2
3  x = np.array([1, 2])
4  y = np.array([4, 2, 1])
5  y = x * y
6  y += 1
```

A) `[1 2 4 2 1 1]`

B) `[5 5 2]`

C) `[5 4 1]`

D) `[-3 11 -6]`

E) `11`

F) `[-4 11 -6]`

G) `[-3, 12, -5]`

H) `[4 4 1 1]`

I) `None`

J) Message d'erreur

Question 3 (1 point)

Que vaut `b` à la fin de l'exécution du code suivant ?

```
1  b = 0
2  for x in [2, 5, 3]:
3    if x > b:
4      b = x
```

A) `10` **B)** `2` **C)** `1.5` **D)** `5` **E)** `1`

F) `253` **G)** `"253"` **H)** `3` **I)** `30` **J)** Message d'erreur

Question 4 (2 points)

Que va afficher le code suivant ?

```
1  def fun(L1, L2):
2    res = []
3    for i in range(len(L1)):
4      d = [L1[i], L2[i]]
5      res.append(d)
6    return res
7
8
9  print(fun([4, 6, 7], [-1, 3, 10]))
```

A) `[3, 9, 17]`

B) `[-1, 3, 10]`

C) `[5, 3, 3]`

D) `[17, 12]`

E) `[]`

F) `[4, -1, 6, 3, 7, 10]`

G) `29`

H) `[[4, -1], [6, 3], [7, 10]]`

I) Message d'erreur

J) Aucune des propositions

Question 5 (2 points)

Que va afficher l'exécution du code suivant ?

```
1  L = [2, 5, 7, 3, 1]
2  x = ""
3  r = []
4  for a in L:
5      x += str(a)
6      r = r + [int(x)]
7  print(r)
```

A) `['2', '5', '7', '3', '1']`

B) `['25731']`

C) `['2', '25', '257', '2573', '25731']`

D) `[]`

E) `'225257257325731'`

F) `[2, 25, 257, 2573, 25731]`

G) `[[2], [25], [257], [2573], [25731]]`

H) Le programme ne s'arrête pas (boucle infinie)

I) Le programme plante (message d'erreur)

Question 6 (3 points)

Dans le code suivant, la fonction `smooth(L)` retourne une nouvelle liste dans laquelle chaque élément est remplacé par la moyenne arithmétique de lui-même, de son prédécesseur et de son successeur dans la liste d'origine. Pour les extrémités, on considère que les éléments en dehors de la liste valent `0`. Malheureusement, le code plante lorqu'on l'exécute.

```
1  def get_value(L, i, default):
2      if i < 0 or i >= len(L):
3          return default
4      else:
5          return L[i]
6
7  def smooth(L):
8      res = []
9      for i in range(len(L)):
10         elem = 0
11         for di in [-1, 0, 1]:
12             elem += get_value(L, i + di)
13         res.append(elem / 3)
14     return res
15
16 print(smooth([2, 1, 6, 2, 1]))
17 # devrait afficher
18 # [1.0, 3.0, 3.0, 3.0, 1.0]
19 # car
20 # [(0+2+1)/3, (2+1+6)/3, (1+6+2)/3, (6+2+1)/3, (2+1+0)/3]
```

Le message d'erreur est le suivant :

```
Traceback (most recent call last):
  File "/Users/lur/exam.py", line 16, in <module>
    print(smooth([2, 1, 6, 2, 1]))
          ~~~~~~^^^^^^^^^^^^^^^^^^
  File "/Users/lur/exam.py", line 12, in smooth
    elem += get_value(L, i + di)
            ~~~~~~~~~^^^^^^^^^^^
TypeError: get_value() missing 1 required positional argument: 'default'
```

Que faut-il changer pour corriger la fonction ? *(les → indiquent le niveau d'indentation)*

A) ligne 1: `def get_value(L, i):`

B) ligne 2: → `if i < 0 and i >= len(L):`

C) ligne 3: → → `return 0`

D) ligne 8: → `res = L`

E) ligne 10: → → `elem = L[i]`

F) lignes 12: → → → `elem += get_value(L, i + di, di)`

G) Les propositions 1 et 3 simultanément

Question 7 (4 points)

Dans le programme suivant, la fonction `find_max_vowels()` devrait renvoyer le mot de la liste reçue en paramètre contenant le plus de voyelles :

```
1  def count_vowels(word):
2      res = 0
3      for c in word:
4          if c in "aeiouy":
5              res += 1
6      return res
7
8  def find_max_vowels(words):
9      count = 0
10     res = None
11     for word in words:
12         if count_vowels(words) > count:
13             res = word
14             count = count_vowels(word)
15     return res
16
17 print(find_max_vowels(["banane", "pomme", "groseille", "cerise", "fraise"]))
```

Le résultat du programme devrait être le suivant:

```
groseille
```

Cependant une erreur s'est glissée dans le programme et on obtient ceci :

```
None
```

Que faut-il changer pour corriger la fonction ? *(les → indiquent le niveau d'indentation)*

A) Ligne 2: → `res = word[0]`

B) Ligne 2: → `res = ""`

C) Ligne 4: → → `if c not in "aeiouy":`

D) Ligne 5: → → → `res += c`

E) Ligne 9: → `count = None`

F) Ligne 10: → `res = 0`

G) Ligne 12: → → `if count_vowels(word) > count:`

H) Ligne 12: → → `if count_vowels(words) > res:`

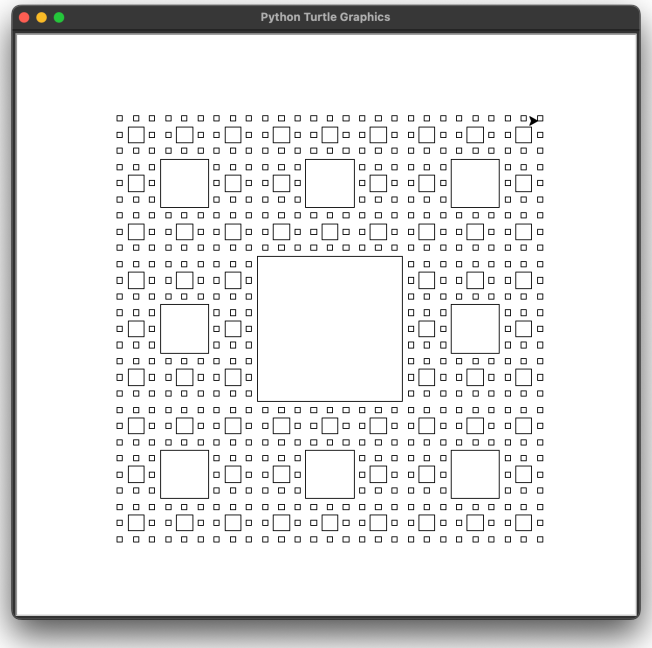
I) Ligne 14: → → `count = count_vowels(words)`

Carpette de Sierpiński

Voici le code d'un programme `turtle` où il manque trois instructions :

```
# première instruction manquante

def go(x, y):
    up()
    goto(x, y)
    down()

def square(x, y, side):
    go(x, y)
    # deuxième instruction manquante
        forward(side)
        left(90)

def carpet(x, y, side):
    third = side / 3
    if third > 5:
        for i in range(3):
            for j in range(3):
                # troisième instruction manquante:
                    square(x + i * third, y + j * third, third)
                else:
                    carpet(x + i * third, y + j * third, third)

carpet(-250, -250, 500)
done()
```

Ce programme devrait afficher la carpette de Sierpiński. Cette figure se construit en divisant un carré en neuf (3 divisions en largeur et en hauteur); on trace le carré du centre et on réapplique la même procédure sur les 8 autres carrés:



Question 8 (2 points)

Dans le programme de la carpette de Sierpiński ci-dessus, quelle proposition choisissez-vous pour la première instruction manquante ? *(Cette instruction n'est pas indentée)*

A) `import turtle`

B) `from pygame import goto, forward, left, done, up, down`

C) `from turtle import goto, forward, right, done, up, down`

D) `from turtle`

E) `from turtle import goto, forward, left, done, up, down`

F) `from turtle import goto, forward, left, up, down`

G) `import pygame`

H) `import goto, forward, left, done, up, down`

Question 9 (2 points)

Dans le programme de la carpette de Sierpiński ci-dessus, quelle proposition choisissez-vous pour la deuxième instruction manquante ? *(Cette instruction est indentée 1 fois)*

A) `for i in range(4):`

B) `while i < 4:`

C) `for i in len(4):`

D) `for i in range(len(4)):`

E) `for i in range(5):`

F) `for i in range(1, 4):`

G) `for i in range(len(side)):`

H) `while side < 4:`

I) `for i in [x, y]:`

J) `for i < 4:`

Question 10 (2 points)

Dans le programme de la carpette de Sierpiński ci-dessus, quelle proposition choisissez-vous pour la troisième instruction manquante ? *(Cette instruction est indentée 4 fois)*

A) `if i == 2 and j == 2:`

B) `if i == j or j == 1:`

C) `if i == 1 or j == 1:`

D) `if i == 1:`

E) `if j == 1:`

F) `if x == 1 and y == 1:`

G) `if x == 1 or y == 1:`

H) `if i == 0 and j == 1:`

I) `if i == 1 or j == 0:`

J) `if i == 1 and j == 1:`

Brouillon