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

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

C) Le type `complex` existe en Python

D) Le langage machine est le langage que comprend le processeur

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

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

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

H) La mémoire RAM est persistante.

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

J) Un SSD est un stockage de masse

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

A) `12`

B) `[-4 11 -6]`

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

D) `[4 4 3 1]`

E) `[1 2 3 4 2 1 1]`

F) `[5 5 4]`

G) `[5 4 3]`

H) `[-3 11 -6]`

I) `None`

J) Message d'erreur

Question 3 (1 point)

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

```
1  b = 42
2  for x in [2, 5, 3]:
3    b += x // 2
```

A) `253` **B)** `121` **C)** `4` **D)** `46` **E)** `47`

F) `8` **G)** `21` **H)** `1.5` **I)** `2.5` **J)** `1`

Question 4 (2 points)

Que va afficher le code suivant ?

```
1  def fun(L1, L2):
2    res = 0
3    for i in range(len(L1)):
4      d = L1[i] - L2[i]
5      if d < 0:
6        res -= d
7      else:
8        res += d
9    return res
10
11
12 print(fun([4, 6, 7, 3], [-1, 3, 10, -5]))
```

A) `0`

B) `[4, 6, 7, 3]`

C) `20`

D) `13`

E) `[5, 3, -3, 8]`

F) `[-1, 3, 10, -5]`

G) `[5, 3, 3, 8]`

H) `19`

I) Message d'erreur

J) Aucune des propositions

Question 5 (2 points)

Que va afficher l'exécution du code suivant ?

```
1  fruits = ['banane', 'pomme', 'cerise', 'fraise', 'groseille']
2  out = []
3  for fruit in fruits:
4    w = ''
5    i = 0
6    while i < len(fruit):
7      l = fruit[i]
8      if l in 'aeiou':
9        w += l
10     i += 1
11   out.append(w)
12 print(out)
```

A) `[]`

B) `['banane', 'pomme', 'cerise', 'fraise', 'groseille']`

C) `['a', 'e', 'i', 'o', 'u']`

D) `['bnn', 'pmm', 'crs', 'frs', 'grsll']`

E) `['w', 'w', 'w', 'w', 'w']`

F) `['aae', 'oe', 'eie', 'aie', 'oeie']`

G) `['lll', 'lll', 'lll', 'lll', 'llllll']`

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 `fun(x, n)` renvoie la liste des divisions de `x` par les `n` premiers diviseurs entiers positifs *(`[5.0, 2.5, 1.6666666666666667, 1.25]` pour l'appel de la ligne 7 car `5.0 = 5/1`, `2.5 = 5/2`, `1.6667 = 5/3` et `1.25 = 5/4`)*. Malheureusement, le code plante lorqu'on l'exécute.

```
1  def fun(x, n):
2    res = []
3    for i in range(n):
4      res.append(x/i)
5    return res
6
7  print(fun(5, 4))
```

Le message d'erreur est le suivant :

```
Traceback (most recent call last):
  File "/Users/lur/Desktop/exo.py", line 7, in <module>
    print(fun(5, 4))
          ~~~^^^^^^
  File "/Users/lur/Desktop/exo.py", line 4, in fun
    res.append(x/i)
               ~^~
ZeroDivisionError: division by zero
```

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

A) ligne 2: → `res = [5.0, 2.5, 1.6666666666666667, 1.25]`

B) ligne 2: → `res = [x/n]`

C) ligne 3: → `for i in range(1, n):`

D) entre les lignes 3 et 4: → → `if i != 0:`

E) ligne 4: → → `res.append(x/(i+1))`

F) entre les lignes 4 et 5: → → `res = [res]`

G) ligne 5: → `return [res]`

H) Les propositions C et G

I) Les propositions D et E

J) Aucune proposition n'est correcte

Question 7 (4 points)

Dans le programme suivant, la fonction `sort()` devrait trier la liste reçue en paramètre :

```
1  def min(L):
2      res = 0
3      for i in range(len(L)):
4          if L[i] < L[res]:
5              res = i
6      return res
7
8  def sort(L):
9      for i in range(len(L)):
10         j = min(L[i:])
11         tmp = L[i]
12         L[i] = L[j]
13         L[j] = tmp
14
15 L = [4, 2, 3, 0, 7]
16 sort(L)
17 print(L)
```

Le résultat du programme devrait être le suivant:

```
[0, 2, 3, 4, 7]
```

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

```
[7, 0, 2, 3, 4]
```

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

A) Ligne 2: → `res = L[0]`

B) Ligne 3: → `for i in range(1, len(L)):`

C) Ligne 4: → → `if i < res:`

D) Ligne 5: → → → `res = L[i]`

E) Ligne 9: → `for i in range(len(L)-1):`

F) Ligne 10: → → `j = min(L[i:]) + i`

G) Ligne 11: → → `tmp = i`

H) Ligne 12: → → `i = j`

I) Ligne 13: → → `j = tmp`

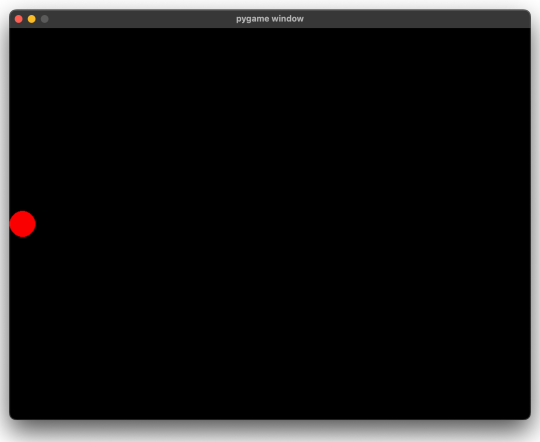
J) Les propositions G, H et I en même temps

Programme `pygame`

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

```
1  import pygame
2  import sys
3
4  pygame.init()
5
6  size = (800, 600)
7  width, height = size
8  radius = 20
9
10 screen = pygame.display.set_mode(size)
11
12 x = radius
13 # Première instruction manquante
14
15 while True:
16     for event in pygame.event.get():
17         if event.type == pygame.QUIT:
18             sys.exit()
19         if event.type == pygame.KEYDOWN:
20             if event.key == pygame.K_RIGHT:
21                 # Deuxième instruction manquante
22         if event.type == pygame.KEYUP:
23             # Troisième instruction manquante
24                 key_pressed = False
25
26     if key_pressed:
27         x += 1
28
29     pygame.draw.rect(screen,(0, 0, 0),pygame.Rect(0, 0, 800, 600))
30
31     pygame.draw.circle(screen, (255, 0, 0), (x, height/2), radius)
32
33     pygame.display.flip()
```

Ce programme devrait afficher un cercle rouge à gauche de l'écran. le cercle devrait bouger vers la droite pendant que la touche « flèche droite » (`K_RIGHT`) est enfoncée.



Voici les propositions pour combler les instructions manquantes *(Adaptez le niveau d'indentation en fonction de l'emplacement)* :

A) `key_pressed = False`

B) `running = False`

C) `key_pressed = event.pos`

D) `if event.key == pygame.K_RIGHT:`

E) `key_pressed = event.key`

F) `if event.key == pygame.KEYDOWN:`

G) `key_pressed = True`

H) `if event.type == pygame.K_RIGHT:`

I) `running = True`

J) `if event.key == pygame.KEYUP:`

Question 8 (2 points)

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

Question 9 (2 points)

Dans le programme `pygame` ci-dessus, quelle proposition choisissez-vous pour la deuxième instruction manquante ? *(Cette instruction est indentée 4 fois)*

Question 10 (2 points)

Dans le programme `pygame` ci-dessus, quelle proposition choisissez-vous pour la troisième instruction manquante ? *(Cette instruction est indentée 3 fois)*

Brouillon