

# 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) Le stockage de masse est plus lent que la mémoire.
- B) la variable `PATH` contient une liste de dossiers
- C) `D:\Document\labo.txt` est un chemin absolu
- D) La mémoire est volatile.
- E) Python est un langage interprété
- F) La commande `ls` permet de changer de répertoire courant
- G) Les instructions exécutées par le processeur sont en langage machine.
- H) Un programme est une suite d'instructions visant à la résolution d'un problème.
- I) `numpy` est un module qui doit être installé pour être utilisé.
- 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 y = np.array([3, 2, 1])
4 y = 2 ** y
5 y *= np.array([2, -1, 1])
```

- A) 15
- B) [18 -4 1]
- C) 10
- D) [2 -1 1]
- E) [1 4 1 3 2 5]
- F) [28 -14 14]
- G) [16 -4 2]
- H) [12 -4 2]
- I) None
- J) Message d'erreur

### Question 3 (2 points)

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

```
1  b = 42
2  c = b % 2 == 0
3  if c and b <= 9 :
4      b -= b / 2
5  else:
6      b = b // 2 ** 2
```

- A) 9                      B) 42                      C) 1                      D) 16                      E) 21
- F) 10.5                      G) 8                      H) 10                      I) 2                      J) 4

### Question 4 (2 points)

Que va afficher le code suivant ?

```
1  def fun(L):
2      if len(L) == 0:
3          return []
4      if L[0] % 2 != 0:
5          return [L[0] / 2] + fun(L[1:])
6      else:
7          return fun(L[1:])
8
9  print(fun([4, 6, 7, 3, 12, -2]))
```

- A) 0                                      B) [4, 6, 7, 3, 12, -2]
- C) [4, 6, 12, -2]                      D) 5.0
- E) [7, 3]                                  F) [2, 3, 6, -1]
- G) [3.5, 1.5]                              H) 10.0
- I) []

### 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 not 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', 'lllll']
- H) Le programme ne s'arrête pas (boucle infinie)
- I) Le programme plante (message d'erreur)

### Question 6 (2 points)

Dans le code suivant, la fonction `fun()` renvoie la liste des sommes de chaque élément avec les éléments suivants de la liste d'entrée  $[6, 1, -7]$  pour l'appel de la ligne 9 car  $6 = 5 + 8 - 7$ ,  $1 = 8 - 7$  et  $-7 = -7$ . Malheureusement, le code plante lorsqu'on l'exécute.

```
1 def fun(L):
2     res = 0
3     for i, elem in enumerate(L):
4         for other in L[i+1:]:
5             elem += other
6         res.append(elem)
7     return res
8
9 print(fun([5, 8, -7]))
```

Le message d'erreur est le suivant :

```
Traceback (most recent call last):
  File "/Users/lur/Desktop/py/truc.py", line 9, in <module>
    print(fun([5, 8, -7]))
    ~~~~~
  File "/Users/lur/Desktop/py/truc.py", line 6, in fun
    res.append(elem)
    ~~~~~
AttributeError: 'int' object has no attribute 'append'
```

Que faut-il changer pour corriger la fonction ? (les  $\rightarrow$  indiquent le niveau d'indentation)

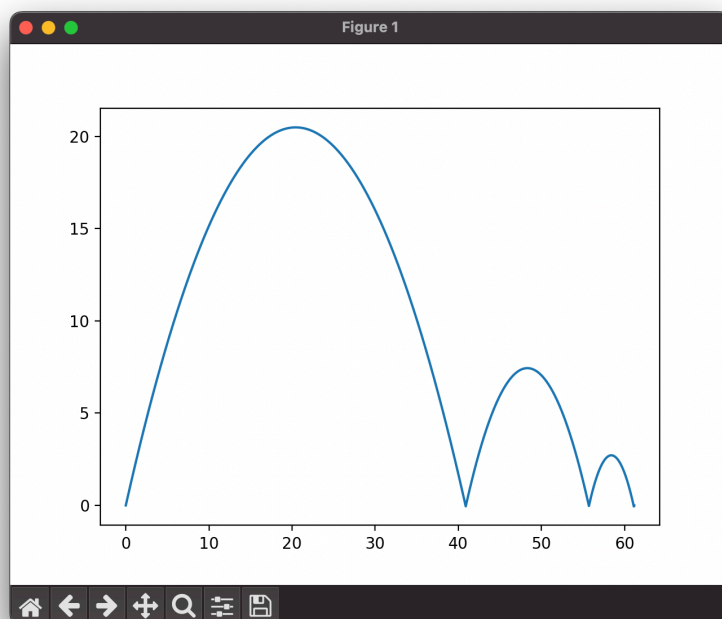
- A) ligne 2:  $\rightarrow$  `res = []`
- B) ligne 3:  $\rightarrow$  `for i, elem in range(len(L)):`
- C) ligne 4:  $\rightarrow \rightarrow$  `for other in L[i+1]:`
- D) entre les lignes 4 et 5:  $\rightarrow \rightarrow \rightarrow$  `res = [res]`
- E) ligne 5:  $\rightarrow \rightarrow \rightarrow$  `elem += [other]`
- F) ligne 6:  $\rightarrow \rightarrow$  `res.append([elem])`
- G) ligne 8:  $\rightarrow$  `return [res]`
- H) Les propositions A et F
- I) Les propositions D et E
- J) Aucune proposition n'est correcte

## Programme mobile ponctuel

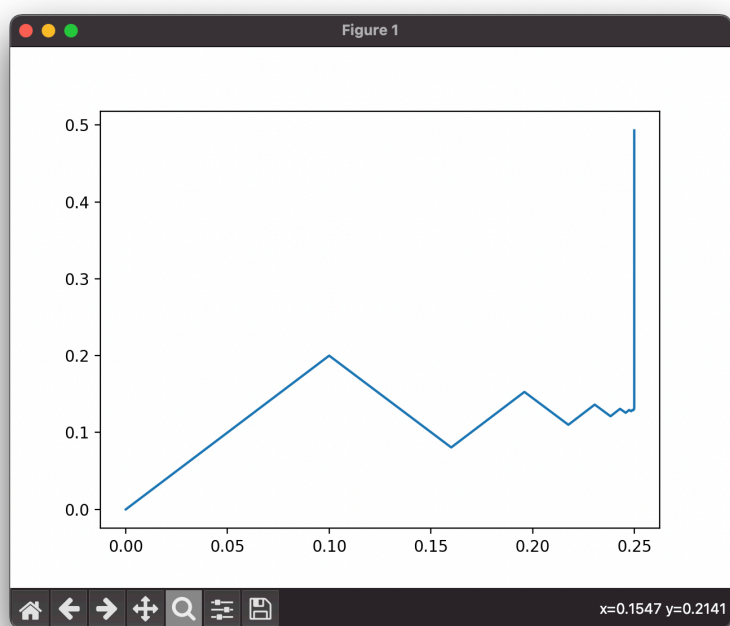
Voici le code d'un programme dessinant la trajectoire entre les instants  $t=0$  et  $t=10$  d'un mobile ponctuel lancé de la position  $(0, 0)$  à l'instant  $t=0$  avec une vitesse initiale  $(10, 20)$  sous l'emprise d'un champs gravitationnel de  $9.81$ . Le mobile rebondit sur le sol ( $y=0$ ) en perdant 40% de sa vitesse à chaque fois.

```
1  from matplotlib import pyplot as plt
2  import numpy as np
3
4  position = np.array([0, 0])
5  velocity = np.array([10, 20])
6
7  # Listes des positions successives du point
8  x = []
9  y = []
10
11 def save_position():
12     x.append(position[0])
13     y.append(position[1])
14
15 g = np.array([0, -9.81])
16 dt = 0.01
17 t = 0
18
19 save_position()
20 while t < 10:
21     position = position + dt * velocity
22     velocity = velocity + dt * g
23
24     # rebond au sol avec une perte de 40% de la vitesse
25     if position[1] > 0:
26         velocity *= np.array([0.6, -0.6])
27
28     save_position()
29     t += dt
30
31 plt.figure()
32 plt.plot(x, y)
33 plt.show()
```

Le résultat de programme devrait être le suivant:



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





### Question 7 (3 points)

Dans le programme du mobile ponctuel, où se situe l'erreur ?

- A) Dans les lignes 1 et 2
- B) Dans les lignes 4 et 5
- C) Dans les lignes 8 et 9
- D) Dans les lignes 15 à 17
- E) Dans la définition de la fonction `save_position()`
- F) Dans la condition du `if` de la ligne 25
- G) Dans le corp du `if` de la ligne 25
- H) À la ligne 29
- I) Dans les lignes 31 à 33
- J) À plusieurs endroits en même temps

### Question 8 (2 points)

Dans le programme du mobile ponctuel, laquelle des variables décrites ci-dessous est globale ?

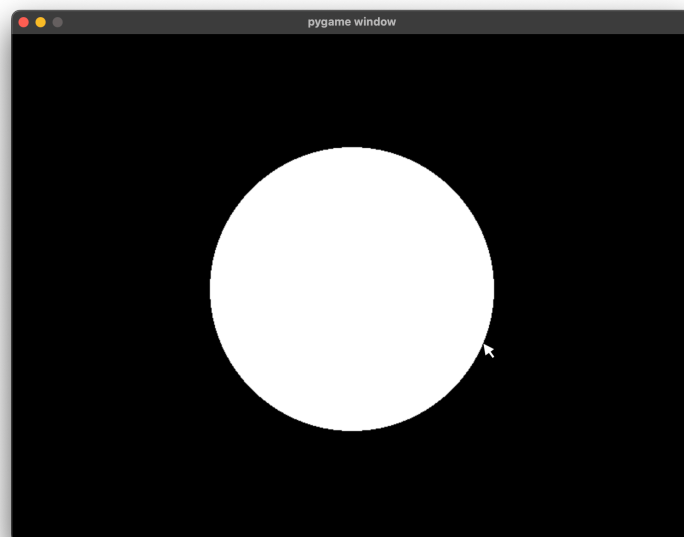
- |  |                                     |
|--|-------------------------------------|
| A) <code>position</code>                               | B) <code>velocity</code>            |
| C) <code>x</code> dans <code>save_position()</code>    | D) <code>g</code>                   |
| E) <code>t</code>                                      | F) <code>dt</code>                  |
| G) <code>y</code> hors de <code>save_position()</code> | H) Toutes les propositions de A à G |
| I) Aucune des propositions                             |                                     |

## Programme pygame

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

```
1  import pygame
2  import sys
3  from math import sqrt
4
5  pygame.init()
6  screen = pygame.display.set_mode((800, 600))
7
8  def distance(p1, p2):
9      return sqrt((p1[0] - p2[0])**2 + (p1[1] - p2[1])**2)
10
11  radius = 0
12  while True:
13      for event in pygame.event.get():
14          if event.type == pygame.QUIT:
15              sys.exit()
16          if event.type == pygame.MOUSEBUTTONDOWN:
17              # première instruction manquante
18
19          pygame.draw.rect(screen,(0, 0, 0),pygame.Rect(0, 0, 800, 600))
20
21          if radius > 0:
22              # deuxième instruction manquante
23
24          pygame.display.flip()
```

Ce programme devrait afficher à chaque clic de souris, un cercle plein, blanc sur fond noir, centré au milieu de la fenêtre et de rayon faisant passé le cercle par le point cliqué.



### Question 9 (2 points)

Dans le programme `pygame` ci-dessus, quelle proposition choisissez-vous pour la première instruction manquante ? (les `→` indiquent le niveau d'indentation)

- A) `→ → → radius = 0`
- B) `→ → → radius = event.pos`
- C) `→ → → radius = distance((400, 300), event.pos)`
- D) `→ → → pygame.draw.circle(screen, (255, 255, 255), event.pos, radius)`
- E) `→ → → radius = distance(event.pos, (0, 0))`
- F) `→ → → points = tuple()`
- G) `→ → → radius = distance((800, 600), (0, 0))`
- H) `→ → → break`
- I) `→ → → radius.append(event.pos)`
- J) `→ → → print('radius')`

### Question 10 (3 points)

Dans le programme `pygame` ci-dessus, quelle proposition choisissez-vous pour la deuxième instruction manquante ? (les `→` indiquent le niveau d'indentation)

- A) `→ → pygame.draw.circle(screen, (255, 255, 255), (800, 600), distance)`
- B) `→ → pygame.draw.circle(screen, (255, 255, 255), (600, 800), radius)`
- C) `→ → continue`
- D) `→ → pygame.draw.circle(screen, (255, 255, 255), (300, 400), radius)`
- E) `→ → radius = distance((400, 300), event.pos)`
- F) `→ → pygame.draw.circle(screen, (255, 255, 255), (800, 600), radius)`
- G) `→ → pygame.draw.circle(screen, (0, 0, 0), (300, 400), radius)`
- H) `→ → pygame.draw.circle(screen, (0, 0, 0), (400, 300), radius)`
- I) `→ → pygame.draw.circle(screen, (255, 255, 255), (0, 0), radius)`
- J) `→ → pygame.draw.circle(screen, (255, 255, 255), (400, 300), radius)`

**Brouillon**