QCM sur la partie programmation.

Question 11. Si `s` est une chaîne de p caractères, combien de caractères comporte la chaîne retournée par l'expression `s[n:m]`, avec $n \leq m \leq p$?

(A) $m - n$

(B) $m - n + 1$

(C) $p - (m - n)$

(D) $m - n - 1$

Question 12. Qu'affiche ce code, si les deux appels `cond1()` et `cond2()` renvoient `False` ?

```
if not cond1():
    print("A")
if cond2():
    print("B")
else:
    print("C")
```

(A)
```
A
B
```

(B)
```
A
```

(C)
```
C
```

(D)
```
A
C
```

Question 13. Qu'affiche ce code sur le terminal ?

```
values = [True, False, True, False, True, False]
i = 0
while values[i]:
    print(i)
    i += 1
```

(A) Rien

(B)
```
0
```

(C)
```
0
2
4
```

(D)
```
1
3
5
```

Question 14. Qu'affiche ce code sur le terminal ?

```
inds = [3, 0, 2, 1]
for i in inds:
    print(inds[i])
```

(A)
```
1
3
2
0
```

(B)
```
3
0
2
1
```

(C)
```
0
1
2
3
```

(D) Une erreur

Question 15. Imaginons une fonction `f` définie comme ci-dessous. Quel appel de cette fonction n'est *pas* possible ?

```
def f(a: int, s: Optional[str], b: float = 0.1) -> None
```

(A) `f(42)`

(B) `f(s="42")`

(C) `f(b=42.0, a=42)`

(D) Aucun de ces trois appels n'est possible.

Question 16. Quelle affirmation sur les fonctions est *fausse* ?

(A) Si une fonction a un autre type de retour que `None`, elle doit livrer une valeur de retour de ce type avec un `return`.

(B) Si une fonction a `None` comme type de retour, elle ne peut pas contenir le mot clé `return`.

(C) Une fonction peut contenir le mot clé `return` plusieurs fois.

(D) Une instruction suivant immédiatement un `return` ne sera pas exécutée vu que le `return` indique la fin de l'exécution du code de la fonction.

Question 17. Quelle expression sous forme de compréhension de liste ci-dessous ne génère *pas* la liste des premières puissances de 2 ?

(A) `[2 ** i for i in range(5)]`

(B) `[n ** i for n in [2] for i in range(0, 5, 1)]`

(C) `[2 ** n for n in [1 for i in range(5)]]`

(D) `[2 ** i for i in [0, 1, 2, 3, 4]]`

Question 18. Si `data` est une liste de `float`s, quelle ligne affichera sur le terminal systématiquement *tous* les éléments de cette liste ?

(A) `print(data[:])`

(B) `print(data)`

(C) `print(data[0:])`

(D) Plusieurs solutions ci-dessus sont correctes

Question 19. Si `f` est une fonction qui accepte un `int` et qui retourne une liste de `int`, quel est son type ?

(A) `Callable[int, List[int]]`

(B) `Callable[[int], List[int]]`

(C) `Callable[[int, int], List]`

(D) `Callable[List, int]`

Question 20. Quelle affirmation sur les threads est incorrecte ?

(A) Un thread peut être créé par un autre thread.

(B) Dans une application à interface graphique (comme avec Tkinter), on doit créer un nouveau thread pour gérer chaque événement (clic de souris, frappe clavier, redimensionnement de la fenêtre, etc.).

(C) Les threads permettent d'exécuter plusieurs séquences d'instructions de manière concurrente (voire vraiment parallèle si la machine le permet).

(D) C'est le système d'exploitation qui décide de quand un thread pourra exécuter son code et de quand il sera obligé de faire une pause.

PARTIE QUESTIONS OUVERTES : RÉPONDEZ SUR LES FEUILLES CI-DESSOUS

Pour ces questions ouvertes, vous n'avez pas besoin d'écrire les **`import`***s.*
Attention à indenter clairement vos blocs selon les lignes verticales !
Utilisez la page 14 si vous n'avez pas assez d'espace sous une question.

Problème 2. (15 points)

a) On veut écrire une classe **`Point`** pour modéliser un point dans le plan, modélisé par deux nombres réels x et y.

- La méthode **`__init__`** est erronée. Corrigez-la.
- Ajoutez la méthode **`__repr__`** (en indiquant aussi le type de retour) pour qu'un point soit transformé en texte selon ce format : **`(1.0, 2.0)`** *(si $x = 1.0$ et $y = 2.0$)*

```
class Point:
      def __init__(x: str, y: str):
            x = self.x
            y = self.y
```

b) Imaginons que $n = 8$ points $P_0, P_1, \ldots, P_7$ sont définis dans une liste ainsi :

```
data_points: List[Point] = [
    Point(1, 7), Point(3, 7), Point(4, 5), Point(5, 10),
    Point(6, 8), Point(9, 10), Point(8, 7), Point(10, 10)
]
```

On veut déterminer les paramètres m et b de la droite d'équation $y = mx + b$ qui est « la plus proche » de cet ensemble de points. On a déterminé comment calculer de « bonnes » valeurs $m^\star$ et $b^\star$, en notant x_i et y_i les deux coordonnées du point P_i :

$$m^\star = \frac{\displaystyle\sum_{i=0}^{n-1} x_i y_i - \frac{1}{n}\sum_{i=0}^{n-1} x_i \sum_{i=0}^{n-1} y_i}{\displaystyle\sum_{i=0}^{n-1} x_i^2 - \frac{1}{n}\left(\sum_{i=0}^{n-1} x_i\right)^2}$$

$$b^\star = \frac{1}{n}\left(\sum_{i=0}^{n-1} y_i - m^\star \sum_{i=0}^{n-1} x_i\right)$$

(suite de la question à la page suivante)

En n'écrivant qu'*une seule boucle*, calculez les valeurs de $m^\star$ et $b^\star$ et stockez-les dans des variables nommées **`m_star`** et **`b_star`**.

c) Écrivez une classe **`Line`** qui modélise une droite dans le plan avec ses deux paramètres m et b. Créez une instance appelée **`best_fit`** en utilisant les valeurs de $m^\star$ et $b^\star$ stockées dans **`m_star`** et **`b_star`**.

d) Complétez, sur la page suivante, la fonction **`mean_squared_error`** qui, à partir d'une droite et d'une liste de points, calcule l'erreur quadratique moyenne MSE, définie ainsi :

$$MSE = \frac{1}{n} \sum_{i=0}^{n-1} \big(y_i - (mx_i + b)\big)^2,$$

où m et b sont les paramètres de la droite passée à la fonction via le paramètre **`line`**.

```
def mean_squared_error(line: ______, points: ____________) -> _______:
```

e) On souhaite vérifier empiriquement avec du code si la droite trouvée est celle qui minimise l'erreur quadratique. Pour ce faire :

- Avec des boucles ou des compréhensions de listes, générez les 16 droites obtenues en choisissant leur paramètre m parmi $\{m^\star - 0.2, m^\star - 0.1, m^\star + 0.1, m^\star + 0.2\}$ et leur paramètre b parmi $\{b^\star - 0.2, b^\star - 0.1, b^\star + 0.1, b^\star + 0.2\}$;
- Pour chacune de ces droites, calculez l'erreur quadratique moyenne avec **`data_points`** en appelant la fonction **`mean_squared_error`** et vérifiez qu'elle ne soit pas inférieure à celle obtenue avec la droite **`best_fit`** ;
- Si aucune des erreurs liées à ces nouvelles droites n'est inférieure, affichez (une seule fois) **`No smaller MSE found`**.

Résultat :

a) (2 pts)	**b)** (4 pts)	**c)** (1 pt)	**d)** (3 pts)	**e)** (5 pts)	**Total** (15 pts)

(Laisser en blanc)