Bac Général Maths + NSI (Mathématiques et Numérique et sciences informatiques)

CORRIGÉ NON OFFICIEL

Proposition de corrigé rédigée par FormaV avec l'aide d'outils d'intelligence artificielle. Elle peut contenir des erreurs et ne remplace ni le corrigé ni le barème officiels de l'examen.

Les sujets officiels des sessions passées et nos sujets d'entraînement sont classés sur https://bac-maths-nsi.fr/annales-sujets-et-corriges

Document mis en ligne par FormaV®. Reproduction ou diffusion sans autorisation interdite.

BACCALAURÉAT GÉNÉRAL • CORRIGÉ DÉTAILLÉ

Numérique et sciences informatiques

Session 2024 • Jour 2

Code sujet **24-NSIG11BIS** 3 heures 30

Proposition de corrigé pédagogique • FormaV

Se repérer dans le corrigé

Exercice 1 — Solitaire bulgare : listes de listes et programmation orientée objet
7 réponses • sujet p. 2–4

Exercice 2 — Architecture réseaux et routage
14 réponses • sujet p. 5–8

Exercice 3 — Programmation Python, traitement de données en table et bases de données relationnelles
11 réponses • sujet p. 9–13

Les numéros, parties et pages renvoient au sujet original. Chaque réponse rappelle la consigne, explique la démarche et donne le résultat.

FormaV - document d'entraînement non officiel

Exercice 1

6 points

Solitaire bulgare : listes de listes et programmation orientée objet

Question 1.

Sujet p. 2

> Vérifier qu'avec 10 cartes, la partie se termine de nouveau avec un état stable.

À chaque étape, si l'état comporte *k* piles, on enlève une carte à chaque pile non vide et l'on forme une nouvelle pile de taille *k*. En appliquant cette règle à 10 cartes :

`[10]` → `[1, 9]` → `[2, 8]` → `[1, 2, 7]` → `[1, 3, 6]` → `[2, 3, 5]` → `[1, 2, 3, 4]`.

Depuis `[1, 2, 3, 4]`, on pioche quatre cartes : la pile de taille 1 disparaît, les trois autres deviennent des piles de tailles 1, 2 et 3, puis on ajoute une pile de taille 4. On retrouve donc `[1, 2, 3, 4]`.

RÉPONSE À RETENIR

Avec 10 cartes, la partie atteint l'état stable `[1, 2, 3, 4]`, qui se réduit sur lui-même.

Question 2.

Sujet p. 3

> Vérifier qu'avec 9 cartes, la partie se termine avec un cycle.

On applique la règle du jeu à partir de l'état initial `[9]` :

`[9]` → `[1, 8]` → `[2, 7]` → `[1, 2, 6]` → `[1, 3, 5]` → `[2, 3, 4]` → `[1, 2, 3, 3]` → `[1, 2, 2, 4]` → `[1, 1, 3, 4]` → `[2, 3, 4]`.

L'état `[2, 3, 4]` est ainsi retrouvé. Comme la règle détermine un unique état suivant pour chaque état, les états recommencent alors à se répéter : la partie forme un cycle.

RÉPONSE À RETENIR

Avec 9 cartes, la partie se termine par le cycle `[2, 3, 4]` → `[1, 2, 3, 3]` → `[1, 2, 2, 4]` → `[1, 1, 3, 4]` → `[2, 3, 4]`.

Question 3.

Sujet p. 3

> Compléter les lignes 15 et 22 de la méthode appelée suivante, qui permet de passer d'un état de la partie à l'état suivant.

À la ligne 15, il faut mémoriser l'état courant avant de calculer son successeur. On peut en ajouter une copie à la liste des états précédents. À la ligne 22, la liste `suiv`, déjà triée à la ligne 21, devient l'état courant.

```
self.precedents.append(self.courant.copy())
...
self.courant = suiv
```

RÉPONSE À RETENIR

Ligne 15 : `self.precedents.append(self.courant.copy())`.
Ligne 22 : `self.courant = suiv`.

FormaV - document d'entraînement non officiel

Question 4.

Sujet p. 3

> Compléter la méthode est_stable, qui renvoie True si l'état courant est le même que le dernier état précédent et False sinon.

Après au moins un appel à `suivante`, le dernier état précédent est le dernier élément de `self.precedents`. On compare cet état à l'état courant. La liste des précédents étant vide au départ, on teste qu'elle n'est pas vide avant d'en lire le dernier élément.

```
def est_stable(self):
    return len(self.precedents) > 0 and self.courant == self.precedents[-1]
```

> **RÉPONSE À RETENIR**
>
> La méthode renvoie `True` si et seulement si la liste des états précédents n'est pas vide et que son dernier état est égal à l'état courant.

Question 5.

Sujet p. 4

> Écrire une fonction partie_stable prenant en paramètre nb_cartes et qui renvoie True si la partie finit par un état stable et False si la partie finit par un état cyclique. On pourra suivre l'algorithme suivant : créer une instance de partie avec le bon nombre de cartes puis, tant que la partie n'est ni stable ni cyclique, passer à l'état suivant.

On crée une partie, puis on passe à l'état suivant tant que l'état courant n'est ni stable ni déjà rencontré. À la sortie de la boucle, on renvoie le résultat du test de stabilité. Celui-ci est évalué en premier dans la condition : si l'état est stable, la fonction renverra bien `True`, même s'il a déjà été rencontré.

```
def partie_stable(nb_cartes):
    partie = Partie(nb_cartes)
    while not partie.est_stable() and not partie.est_cyclique():
        partie.suivante()
    return partie.est_stable()
```

Pour un nombre initial positif de cartes, la somme des tailles des piles reste égale à `nb_cartes`. Il n'existe qu'un nombre fini d'états possibles : ce sont des listes de tailles positives dont la somme vaut ce nombre. La partie finit donc par atteindre un état stable ou par retrouver un état déjà rencontré, ce qui arrête la boucle.

> **RÉPONSE À RETENIR**
>
> La fonction renvoie `True` si la partie atteint un état stable et `False` si elle aboutit à un cycle non stable.

Question 6.

Sujet p. 4

> Proposer des tests pour vérifier si votre fonction partie_stable semble correcte.

On peut tester les deux issues possibles à l'aide des exemples vérifiables dans le sujet : 10 et 15 cartes conduisent à un état stable, tandis que 9 et 12 cartes conduisent à un cycle. Une carte fournit aussi un petit test : `[1]` devient `[1]` après une étape, donc l'état est stable.

FormaV - document d'entraînement non officiel

```
assert partie_stable(1) is True
assert partie_stable(9) is False
assert partie_stable(10) is True
assert partie_stable(12) is False
assert partie_stable(15) is True
```

RÉPONSE À RETENIR

Les tests donnent `True` pour 1, 10 et 15 cartes, et `False` pour 9 et 12 cartes, conformément aux états décrits dans le sujet.

Question 7.

Sujet p. 4

> Dans la méthode suivante, on trie une liste qui est déjà triée sauf éventuellement la valeur len(valeurs). Donner, en le justifiant, un tri classique qui serait plus efficace dans ce cas.

Dans la construction de `suiv`, l'élément `len(valeurs)` est placé en premier, puis les autres valeurs sont ajoutées dans l'ordre croissant. Ainsi, `suiv[1:]` est déjà triée ; seul le premier élément peut être mal placé. Un tri par insertion, qui insère ce premier élément à sa place dans la partie triée, suffit. La recherche et l'insertion nécessitent au plus un nombre d'opérations proportionnel à la longueur de la liste, alors qu'un tri général ne profite pas de l'ordre déjà présent.

```
def tri_insertion_exceptionnel(suiv):
    valeur = suiv[0]
    i = 1
    while i < len(suiv) and suiv[i] < valeur:
        i += 1
    return suiv[1:i] + [valeur] + suiv[i:]
```

Cette fonction suppose, comme dans le cas de la question, que `suiv[1:]` est triée ; elle renvoie la liste complète dans l'ordre croissant.

RÉPONSE À RETENIR

Le tri par insertion de l'unique valeur éventuellement mal placée est adapté : il trie cette liste en temps linéaire en sa longueur.

Exercice 2

6 points

Architecture réseaux et routage

Question 1. — 192.168.129.3

Sujet p. 5

> Indiquer à quel réseau (A, B, C, D ou AUCUN) appartient l'adresse de poste de travail 192.168.129.3. Les adresses de réseau et masques des réseaux A, B, C et D sont indiqués dans le sujet.

Le réseau D a pour adresse 192.168.128.0 et pour masque 255.255.254.0. Ce masque couvre les adresses dont le troisième octet vaut 128 ou 129. L'adresse 192.168.129.3 appartient donc au réseau D.

RÉPONSE À RETENIR

D

Question 1. — 192.168.194.2

Sujet p. 5

> Indiquer à quel réseau (A, B, C, D ou AUCUN) appartient l'adresse de poste de travail 192.168.194.2. Les adresses de réseau et masques des réseaux A, B, C et D sont indiqués dans le sujet.

Les réseaux B et C couvrent respectivement les adresses 192.168.192.0 à 192.168.192.255 et 192.168.193.0 à 192.168.193.255. L'adresse 192.168.194.2 ne se trouve dans aucun de ces réseaux. Elle ne correspond pas non plus aux plages couvertes par A ou D.

RÉPONSE À RETENIR

AUCUN

Question 1. — 192.168.192.7

Sujet p. 5

> Indiquer à quel réseau (A, B, C, D ou AUCUN) appartient l'adresse de poste de travail 192.168.192.7. Les adresses de réseau et masques des réseaux A, B, C et D sont indiqués dans le sujet.

Le réseau B a pour adresse 192.168.192.0 et pour masque 255.255.255.0 : ses adresses de postes sont dans la plage 192.168.192.1 à 192.168.192.254. L'adresse donnée est dans cette plage.

RÉPONSE À RETENIR

B

Question 1. — 192.168.72.25

Sujet p. 5

> Indiquer à quel réseau (A, B, C, D ou AUCUN) appartient l'adresse de poste de travail 192.168.72.25. Les adresses de réseau et masques des réseaux A, B, C et D sont indiqués dans le sujet.

Le masque 255.255.248.0 du réseau A couvre les troisièmes octets de 64 à 71. Le troisième octet de 192.168.72.25 vaut 72, donc cette adresse n'appartient pas à A. Elle n'appartient ni à B, ni à C, ni à D.

RÉPONSE À RETENIR

AUCUN

FormaV - document d'entraînement non officiel

Question 2.

Sujet p. 5

> Indiquer, pour les réseaux A et D, le nombre de machines qu'ils peuvent effectivement contenir au maximum.

Pour le réseau A, le masque 255.255.248.0 correspond à 21 bits de réseau, donc 11 bits d'hôte. Il y a en tout 2^{11} adresses, dont l'adresse de réseau et l'adresse de diffusion ne sont pas attribuables à des machines : $2^{11} - 2 = 2046$.

Pour le réseau D, le masque 255.255.254.0 correspond à 23 bits de réseau, donc 9 bits d'hôte. Le nombre maximal de machines est $2^9 - 2 = 510$.

RÉPONSE À RETENIR

Réseau A : **2046 machines**. Réseau D : **510 machines**.

Question 3.

Sujet p. 5, 6

> Choisir les deux adresses IP manquantes : celle de R1 dans le réseau A et celle de R3 dans le réseau B. Prendre la plus petite adresse IP disponible qui convienne à chaque fois.

Dans le réseau A, R2 possède l'adresse 192.168.64.2. La plus petite adresse disponible pour l'interface de R1 est donc 192.168.64.1.

Dans le réseau B, R2 possède l'adresse 192.168.192.1. La plus petite adresse disponible pour R3 est donc 192.168.192.2.

RÉPONSE À RETENIR

R1 dans A : **192.168.64.1**. R3 dans B : **192.168.192.2**.

Question 4.

Sujet p. 6, 7

> Compléter la table de routage de R2.

R2 est directement connecté aux réseaux A et B : leurs passerelles sont 0.0.0.0 et leur nombre de sauts est 0. Pour atteindre C, R2 passe par R3, dont l'adresse dans B est 192.168.192.2 : la route compte 1 saut. Pour atteindre D, il passe par R1, dont l'adresse dans A est 192.168.64.1 : cette route compte également 1 saut.

RÉPONSE À RETENIR

Réseau	Masque	Passerelle	Hops/Sauts
192.168.64.0	255.255.248.0	0.0.0.0	0
192.168.192.0	255.255.255.0	0.0.0.0	0
192.168.193.0	255.255.255.0	192.168.192.2 (R3)	1
192.168.128.0	255.255.254.0	192.168.64.1 (R1)	1

FormaV - document d'entraînement non officiel

Question 5.

Sujet p. 6, 7

> Après que chaque routeur aura communiqué une seule fois sa table de routage aux routeurs voisins, le routeur R4 aura reçu les tables de routage de R1 et R3. Donner la nouvelle table de routage de R4 après ces échanges, en présentant les tables comme aux questions précédentes.

R4 est initialement directement connecté à C et à D : il conserve donc ces routes à 0 saut. En ajoutant 1 saut aux distances annoncées par ses voisins, il peut atteindre A en 1 saut par R1 et B en 1 saut par R3. Les annonces reçues pour C et D proposent des routes moins intéressantes que les routes directement connectées à R4. Les passerelles sont les adresses de R1 et R3 sur les réseaux partagés D et C.

RÉPONSE À RETENIR

Réseau	Masque	Passerelle	Hops/Sauts
192.168.64.0	255.255.248.0	192.168.128.1 (R1)	1
192.168.192.0	255.255.255.0	192.168.193.1 (R3)	1
192.168.193.0	255.255.255.0	0.0.0.0	0
192.168.128.0	255.255.254.0	0.0.0.0	0

Question 6. — 192.168.193.17

Sujet p. 6, 7, 8

> Le poste 192.168.192.17 essaie de joindre le nœud 192.168.193.17. Indiquer quels seront les routeurs traversés, dans l'ordre, par les datagrammes IP.

Le poste 192.168.192.17 est dans B, mais 192.168.193.17 est dans C. Le poste envoie donc le datagramme à sa passerelle par défaut, R2. D'après la table de R2, la route vers C passe par R3, qui est directement connecté à C.

RÉPONSE À RETENIR

Les routeurs traversés, dans l'ordre, sont **R2 puis R3**.

Question 6. — 192.168.129.17

Sujet p. 6, 7, 8

> Le poste 192.168.192.17 essaie de joindre le nœud 192.168.129.17. Indiquer quels seront les routeurs traversés, dans l'ordre, par les datagrammes IP.

L'adresse 192.168.129.17 appartient à D. Le poste envoie le datagramme à R2, sa passerelle par défaut. R2 atteint D en passant par R1, qui est directement connecté à D.

RÉPONSE À RETENIR

Les routeurs traversés, dans l'ordre, sont **R2 puis R1**.

Question 6. — 192.168.65.17

Sujet p. 6, 7, 8

> Le poste 192.168.192.17 essaie de joindre le nœud 192.168.65.17. Indiquer quels seront les routeurs traversés, dans l'ordre, par les datagrammes IP.

L'adresse 192.168.65.17 appartient au réseau A, car le masque de A couvre les troisièmes octets de 64 à 71. Le poste, qui n'est pas lui-même dans A, transmet le datagramme à R2. Le réseau A étant directement connecté à R2, aucun autre routeur n'est traversé.

RÉPONSE À RETENIR

Le seul routeur traversé est **R2**.

Question 6. — 192.168.192.27

Sujet p. 6, 7, 8

> Le poste 192.168.192.17 essaie de joindre le nœud 192.168.192.27. Indiquer quels seront les routeurs traversés, dans l'ordre, par les datagrammes IP.

Les adresses 192.168.192.17 et 192.168.192.27 appartiennent toutes deux au réseau B. Le poste considère donc la destination comme directement joignable sur son réseau local et lui envoie le datagramme sans passer par sa passerelle.

RÉPONSE À RETENIR

Aucun routeur n'est traversé.

Question 7.

Sujet p. 8

> Le réseau D doit être découpé en deux sous-réseaux de tailles égales. Indiquer l'adresse IP et le masque associés à chacun des deux sous-réseaux.

Le réseau D, de masque 255.255.254.0, couvre les troisièmes octets 128 et 129. Le découper en deux sous-réseaux de même taille revient à le diviser en deux réseaux /24.

RÉPONSE À RETENIR

Adresse de réseau	Masque
192.168.128.0	255.255.255.0
192.168.129.0	255.255.255.0

Question 8.

Sujet p. 7, 8

> De ces deux sous-réseaux, celui dont le troisième octet est le plus faible est relié à R1 et R4. L'autre est uniquement relié à R1. En supposant que les tables de routage aient été retouchées à la suite de cette modification, indiquer quelles seront les nouvelles tables de routage de R1 et R4 après qu'elles auront été stabilisées par le protocole RIP. Si des adresses IP manquent, choisir des adresses possibles.

Après le découpage, R1 est directement connecté à A, à 192.168.128.0/24 et à 192.168.129.0/24. R4 est directement connecté à C et à 192.168.128.0/24. On choisit des adresses d'interface possibles : R1 garde 192.168.128.1 sur le sous-réseau partagé avec R4, R4 y utilise 192.168.128.2, et R4 garde 192.168.193.2 dans C. On choisit aussi 192.168.129.1 pour R1 dans le sous-réseau relié uniquement à lui.

FormaV - document d'entraînement non officiel

Après stabilisation de RIP, R1 atteint B par R2 en 1 saut et C par R4 en 1 saut. Les deux sous-réseaux issus de D lui sont directement connectés. R4 atteint A par R1 en 1 saut, B par R3 en 1 saut et 192.168.129.0/24 par R1 en 1 saut ; C et 192.168.128.0/24 lui sont directement connectés.

RÉPONSE À RETENIR

Table de routage de R1

Réseau	Masque	Passerelle	Hops/Sauts
192.168.64.0	255.255.248.0	0.0.0.0	0
192.168.192.0	255.255.255.0	192.168.64.2 (R2)	1
192.168.193.0	255.255.255.0	192.168.128.2 (R4)	1
192.168.128.0	255.255.255.0	0.0.0.0	0
192.168.129.0	255.255.255.0	0.0.0.0	0

Table de routage de R4

Réseau	Masque	Passerelle	Hops/Sauts
192.168.64.0	255.255.248.0	192.168.128.1 (R1)	1
192.168.192.0	255.255.255.0	192.168.193.1 (R3)	1
192.168.193.0	255.255.255.0	0.0.0.0	0
192.168.128.0	255.255.255.0	0.0.0.0	0
192.168.129.0	255.255.255.0	192.168.128.1 (R1)	1

Exercice 3

8 points

Programmation Python, traitement de données en table et bases de données relationnelles

Partie A

Question 1.

Sujet p. 9, 10

> L'altitude du premier point est '1457'. Cette chaîne peut être affichée ainsi : print(tab_trace[0][2]). Donner l'instruction permettant d'afficher la longitude du dernier point uniquement.

Dans chaque enregistrement, la longitude est à l'indice `1`. L'indice `-1` désigne le dernier enregistrement de la liste, quelle que soit sa longueur.

RÉPONSE À RETENIR

```
print(tab_trace[-1][1])
```

Cette instruction affiche uniquement la longitude du dernier point.

Question 2.

Sujet p. 9, 10

> Compléter les lignes 6 à 8 de la fonction creation_trace. Cette fonction prend en paramètre une liste similaire à tab_trace et renvoie une liste de dictionnaires contenant chacun une position horodatée, avec les quatre clés lat, lon, alt et tsp. Les latitudes et longitudes sont des nombres à virgule flottante, et les altitudes et timestamps des entiers.

Les champs de chaque enregistrement sont, dans l'ordre, latitude, longitude, altitude et timestamp. On convertit les deux coordonnées avec `float` et l'altitude ainsi que le timestamp avec `int`, avant d'ajouter le dictionnaire à la liste.

RÉPONSE À RETENIR

```
def creation_trace(tab_trace):
    trace = []
    for enregistrement in tab_trace:
        pt = {}
        pt['lat'] = float(enregistrement[0])
        pt['lon'] = float(enregistrement[1])
        pt['alt'] = int(enregistrement[2])
        pt['tsp'] = int(enregistrement[3])
        trace.append(pt)
    return trace
```

Question 3.

Sujet p. 10, 11

> Compléter la fonction valeurs_altitude, qui prend en paramètre une trace et renvoie la liste des altitudes successives. Elle doit satisfaire les assertions données pour les quatre premiers éléments de l'exemple : 1457, 1447, 1440 et 1439.

Chaque point est un dictionnaire dont l'altitude est associée à la clé `'alt'`. On parcourt la trace dans l'ordre, on récupère chaque altitude et on la place dans une nouvelle liste.

RÉPONSE À RETENIR

```
def valeurs_altitude(trace):
    altitudes = []
    for point in trace:
        altitudes.append(point['alt'])
    return altitudes
```

Pour les quatre premiers points de l'exemple, la liste commence par `[1457, 1447, 1440, 1439]`, conformément aux assertions.

Question 4.

Sujet p. 11

> Donner le code de la fonction denivele, qui prend en paramètre une liste d'altitudes et renvoie un tuple contenant le dénivelé cumulé positif et le dénivelé cumulé négatif, qui sont tous deux des nombres positifs. La fonction doit renvoyer (90, 70) pour denivele([1400, 1420, 1450, 1450, 1440, 1390, 1380, 1420]).

Pour chaque paire d'altitudes consécutives, on calcule la variation « altitude suivante moins altitude courante ». Une variation positive augmente le cumul positif ; une variation négative augmente le cumul négatif de sa valeur absolue ; une variation nulle ne contribue à aucun cumul.

Pour la liste donnée, les variations sont `+20, +30, 0, -10, -50, -10, +40`. Ainsi, le cumul positif vaut `20 + 30 + 40 = 90` et le cumul négatif vaut `10 + 50 + 10 = 70`.

RÉPONSE À RETENIR

```
def denivele(altitudes):
    denivele_positif = 0
    denivele_negatif = 0
    for i in range(len(altitudes) - 1):
        variation = altitudes[i + 1] - altitudes[i]
        if variation > 0:
            denivele_positif += variation
        elif variation < 0:
            denivele_negatif += -variation
    return (denivele_positif, denivele_negatif)
```

La fonction renvoie bien `(90, 70)` pour la liste fournie.

Question 5.

Sujet p. 10, 11, 12

> Donner le code de la fonction asc_desc. Elle prend en paramètre une trace (liste renvoyée par creation_trace) et renvoie un tuple de trois valeurs entières contenant, dans cet ordre, le nombre de secondes d'ascension, le nombre

FormaV - document d'entraînement non officiel

de secondes de descente et le nombre de secondes sur plat. L'exemple fourni donne (190, 233, 22).

Chaque intervalle entre deux points est classé selon la variation d'altitude entre ces points. Sa durée est la différence entre les deux timestamps. On ajoute cette durée au cumul d'ascension si l'altitude augmente, au cumul de descente si elle diminue, et au cumul de plat si elle ne change pas. Les timestamps étant des entiers en secondes, les cumuls obtenus sont des entiers.

RÉPONSE À RETENIR

```
def asc_desc(trace):
    temps_ascension = 0
    temps_descente = 0
    temps_plat = 0

    for i in range(len(trace) - 1):
        duree = trace[i + 1]['tsp'] - trace[i]['tsp']
        variation = trace[i + 1]['alt'] - trace[i]['alt']

        if variation > 0:
            temps_ascension += duree
        elif variation < 0:
            temps_descente += duree
        else:
            temps_plat += duree

    return (temps_ascension, temps_descente, temps_plat)
```

Le résultat annoncé pour la trace de l'exemple est `(190, 233, 22)` secondes.

Question 6.

Sujet p. 10, 12

Écrire une fonction nommée distance(p1: dict, p2: dict) qui prend en paramètre deux points (deux dictionnaires) et renvoie la distance qui les sépare en mètres à la surface du globe, sans tenir compte de l'altitude. Utiliser la formule fournie, avec un rayon terrestre R de 6 371 000 m et les angles exprimés en radians.

La formule donne la distance à la surface du globe à partir des latitudes et longitudes. Ces coordonnées étant stockées en degrés, on les convertit en radians avant d'utiliser les fonctions trigonométriques. On multiplie ensuite l'angle obtenu par le rayon terrestre de `6371000` mètres. Le bornage de l'argument de `acos` entre `-1` et `1` protège contre de légères erreurs d'arrondi flottant.

RÉPONSE À RETENIR

```
import math

def distance(p1: dict, p2: dict):
    lat1 = math.radians(p1['lat'])
    lat2 = math.radians(p2['lat'])
    lon1 = math.radians(p1['lon'])
    lon2 = math.radians(p2['lon'])

    argument = (
        math.sin(lat1) * math.sin(lat2)
        + math.cos(lat1) * math.cos(lat2) * math.cos(lon1 - lon2)
    )
    argument = max(-1.0, min(1.0, argument))
    return 6371000 * math.acos(argument)
```

La distance renvoyée est exprimée en mètres et ne tient pas compte des altitudes.

Question 7.

Sujet p. 12

> En utilisant le théorème de Pythagore, proposer le code d'une fonction distance_precise(p1: dict, p2: dict) qui tient aussi compte de l'altitude des deux points pour le calcul de la distance, comme sur la figure 2.

La distance calculée par `distance` représente la distance horizontale à la surface terrestre. Avec la différence d'altitude entre les deux points, elle forme les deux côtés perpendiculaires d'un triangle rectangle. Le théorème de Pythagore donne donc la distance en tenant compte de l'altitude.

RÉPONSE À RETENIR

```
import math

def distance_precise(p1: dict, p2: dict):
    distance_horizontale = distance(p1, p2)
    difference_altitude = p2['alt'] - p1['alt']
    return math.sqrt(
        distance_horizontale ** 2 + difference_altitude ** 2
    )
```

Le signe de la différence d'altitude n'a pas d'effet sur le résultat, car cette différence est mise au carré.

Partie B

Question 8.

Sujet p. 13

> Donner le résultat de la requête SELECT lieu, dist FROM trails WHERE denivele_p > 1000; lorsque la relation ne contient que les quatre enregistrements présentés dans le tableau.

La condition `denivele_p > 1000` est vérifiée par les lignes dont le dénivelé positif vaut `1919` et `10000`. La requête affiche seulement les colonnes `lieu` et `dist`.

RÉPONSE À RETENIR

lieu	dist
Mont Blanc	21.3
Mont Blanc	171

Question 9.

Sujet p. 13

> Donner la requête SQL permettant d'obtenir uniquement les id et les lieux (colonne lieu) des trails dont la distance dépasse 50 km.

On sélectionne uniquement les colonnes `id` et `lieu`, puis on filtre les courses dont la distance est strictement supérieure à `50` km.

RÉPONSE À RETENIR

```
SELECT id, lieu
FROM trails
WHERE dist > 50;
```

Sur les quatre lignes de l'échantillon, les distances qui dépassent 50 km sont `171` et `92` :

id	lieu
2	Mont Blanc
6	Ile de Ré

Question 10.

Sujet p. 13

> On suppose que si les dénivelés positifs et négatifs sont égaux, alors la course est une balade parfaite. En utilisant cette idée, donner une requête SQL permettant d'afficher uniquement le lieu et la distance de toutes les courses qui ne sont pas des balades parfaites.

Une balade parfaite vérifie l'égalité entre les dénivelés positif et négatif. Les courses qui ne sont pas des balades parfaites vérifient donc leur inégalité. L'opérateur SQL standard <> signifie « différent de ».

RÉPONSE À RETENIR

```
SELECT lieu, dist
FROM trails
WHERE denivele_p <> denivele_n;
```

Sur les quatre lignes de l'échantillon, seule la première a des dénivelés différents :

lieu	dist
Mont Blanc	21.3

FormaV - document d'entraînement non officiel

Question 11.

Sujet p. 13

> Donner la requête SQL affichant la distance cumulée de tous les trails dont le dénivelé cumulé positif est supérieur à 1 000 m. Sur l'échantillon du tableau, la réponse serait 192.3.

La requête filtre les courses dont le dénivelé cumulé positif est strictement supérieur à `1000` mètres, puis additionne leurs distances avec `SUM`. Dans l'échantillon, les lignes retenues ont des distances de `21.3` km et `171` km.

RÉPONSE À RETENIR

```
SELECT SUM(dist)
FROM trails
WHERE denivele_p > 1000;
```

La somme vaut `21.3 + 171 = 192.3` km.

SUM(dist)
192.3

Sujet : https://www.sujetdebac.fr/annales-pdf/2024/spe-numerique-informatique-2024-suede-2-sujet-officiel.pdf
© FormaV EI • Proposition de correction, distincte d'un corrigé officiel.

Copyright © 2027 FormaV®. Tous droits réservés.

FormaV® est une marque déposée auprès de l'INPI sous le numéro 5227897.

Ce document a été élaboré par FormaV® avec le plus grand soin afin d'accompagner chaque apprenant vers la réussite de ses examens. Son contenu (textes, graphiques, méthodologies, tableaux, exercices, concepts, mises en forme) constitue une œuvre protégée par le droit d'auteur.

Toute copie, partage, reproduction, diffusion ou mise à disposition, même partielle, gratuite ou payante, est strictement interdite sans accord préalable et écrit de FormaV®, conformément aux articles L.111-1 et suivants du Code de la propriété intellectuelle. Dans une logique anti-plagiat, FormaV® se réserve le droit de vérifier toute utilisation illicite, y compris sur les plateformes en ligne ou sites tiers.

En utilisant ce document, vous vous engagez à respecter ces règles et à préserver l'intégrité du travail fourni. La consultation de ce document est strictement personnelle.

Merci de respecter le travail accompli afin de permettre la création continue de ressources pédagogiques fiables et accessibles.