Sujets retenus

Les sujets suivants, au nombre de 21, sont d'une difficulté assez homogène et couvrent de nombreuses notions du programme du cycle terminal en NSI. Nous proposons donc de retenir cette liste pour le choix de l'épreuve pratique de NSI le Vendredi 20 Juin 2025.

Il est possible de donner le même sujet aux 4 candidats ou de donner 4 sujets différents à chacun des candidats au cours de la même session. Un système de tirage au sort peut également être mis en œuvre.

Liste des sujets :

- $1 - 4 - 6 - 7$
- $12 - 15 - 17$
- $20 - 21 - 23 - 28 - 29$
- $30 - 33 - 35 - 36 - 37$
- $41 - 43 - 45 - 47$

Proposition de barème

La session 2025 des épreuves pratiques propose comme l'an dernier une notation sur 20 points. Afin d'aider les examinateurs lors de leurs évaluations des candidats, nous avons réfléchi à une proposition de barème.

Il sera tenu compte dans la notation de l'ampleur de l'aide apportée par l'examinateur au candidat

Exercice 1

- Questions : 3 points
- Programme : 7 points. Plusieurs cas peuvent se présenter :
  - Le programme ne passe pas les tests proposés : entre 0 et 3 points
  - Le programme n'est pas correct (par exemple, mauvais traitement des cas limites) mais passe les tests proposés : 4 points
  - Le programme est correct et passe les tests proposés : 5 points
  - Le programme est correct, lisible et passe les tests proposés : 6 points
  - Le programme est correct, lisible, passe les tests proposés et est efficace en termes de coût : 7 points

Exercice 2

- Questions : 4 points
- Programme : 6 points. Plusieurs cas peuvent se présenter :
  - Le programme ne passe pas les tests proposés : entre 0 et 4 points
  - Le programme n'est pas correct (par exemple, mauvais traitement des cas limites) mais passe les tests proposés : 5 points
  - Le programme est correct et passe les tests proposés : 6 points

Questions aux candidats

Les questions qui suivent ne sont en rien limitatives. Il s'agit d'entrées classiques que nous utilisons régulièrement dans nos pratiques personnelles dans l'enseignement de la NSI.

Fonctions

1. Quels sont les paramètres de la fonction ? Quels sont leurs types ?
2. Quelles sont les données en sortie de la fonction ? Quels sont leurs types ?
3. Ecrire une docstring (documentation) pour la fonction.
4. Quelle est la différence entre paramètres et arguments pour une fonction ?
5. La fonction est-elle récursive ? Pourquoi ? Quels sont les cas d'arrêts ? Quelle est son coût ?
6. Pour une fonction récursive, comment peut-on minimiser les risques de dépassement de pile ?
7. Quel est le coût de la fonction ? Existe-t-il des moyens de le réduire ?

Boucles

8. Dans le cas d'un `for`, de quel type de parcours s'agit-il (indice/élément) et pourquoi avoir choisi ce parcours ?
9. Dans le cas d'une boucle `while`, comment peut-on s'assurer qu'elle se termine ?
10. Quel autre type de boucle existe-t-il ? Pourquoi avez-vous utilisé ce type de boucle (entre `for` et `while`) ?
11. Donner un exemple de situation où une boucle `for` serait préférable à une boucle `while`, et vice-versa.
12. Comment les opérations dans la boucle affectent-elles la performance globale de votre programme ?

Instructions conditionnelles

13. Expliquer la ligne de l'instruction conditionnelle. Quelle est la différence entre `==` et `=` ?
14. Quel est le type de l'expression `a == b` ?

Variables

15. Quel est l'utilité de cette variable ? Pourquoi avoir choisi ce nom de variable ? En existe-t-il un plus pertinent ?
16. Quel est le type de cette variable ? Pourquoi avoir fait ce choix ?
17. La valeur de cette variable est-elle accessible en dehors de la fonction ? Pourquoi ?

Opérations

18. Quelle est la différence entre `//` et `/` ?
19. Quelle est la différence entre `*` et `**` ?
20. Pourquoi utiliserait-on `//` pour des calculs impliquant des indices ou des tailles en programmation ?
21. Pourquoi utiliserait-on `%` pour des calculs impliquant des indices ou des tailles en programmation ?
22. Dans quel contexte `*` pourrait-il être particulièrement utile en dehors des mathématiques ?

Structures de données

23. Comment obtenir le nombre d'éléments de cette structure ? Comment accéder à un élément ? Comment ajouter un élément ? Comment modifier un élément ? Est-ce que c'est toujours possible ?
24. En quoi cette structure (liste python, dictionnaire, pile, file, arbre, graphe, etc.) est-elle adaptée à l'exercice ?
25. Expliquer la différence entre une liste Python et un tuple ? Dans quel cas utiliser l'un plutôt que l'autre ?
26. Réécrire la construction de cette liste Python / dictionnaire par compréhension ?
27. Pour le parcours de ce dictionnaire, quelles données sont parcourues (clef, valeur ou couple cle:valeur) ?

Programmation défensive, correction, validité

28. Ecrire des assertions pour tester les postconditions de la fonction.
Envisager les cas particuliers (liste vide, liste unitaire, flottants, doublons, nombres négatifs, etc.).
29. Ecrire d'autres tests du comportement attendu de la fonction.
30. Ecrire un test qui provoque une erreur.
31. Ecrire des assertions sur les préconditions. Où les placer dans le code ?
32. Comment fonctionne le programme ? À quoi sert cette ligne ? Que se passe-t-il si on supprime cette ligne ?

Culture générale

33. Est-ce que toute boucle bornée peut être transformée en boucle non bornée ? Si oui, comment ?
34. Est-ce que toute boucle non bornée peut être transformée en boucle bornée ? Non, exemple…
35. Comment obtenir la docstring (documentation) d'une fonction ? (Réponse : la fonction **`help`**)
36. Peut-on nommer une fonction ou une variable **`max`** ? Est-ce un nom réservé, comme **`len`**, **`append`**, etc. ?
37. Comment appelle-t-on le fait d'écrire **`i = i + 1`** ?
38. Quelle différence faites-vous entre **`sorted(tab)`** et **`tab.sort()`**, où **`tab`** est une liste Python ?
39. Quelle différence faites-vous entre **`tab.append(i)`** et **`tab = tab + [i]`**, où **`tab`** est une liste Python ?

Programmation Objet

40. Quel est le constructeur de cette classe ?
41. Quels sont les attributs et les méthodes ?
42. Que représente **`self`** ?
43. Quel est l'intérêt de la programmation Objet ?

Débogage et analyse de code

44. Comment procédez-vous pour déboguer votre code ? Utilisez-vous des outils spécifiques (comme des débogueurs ou des impressions) pour identifier et résoudre les problèmes ?

Développement et maintenabilité

45. Comment assurer que votre code soit lisible et maintenable pour d'autres développeurs ou pour vous ?

Tests et documentation

46. Quelle est l'importance de la documentation de votre code ?
Quels éléments incluez-vous généralement dans votre documentation ?

Réflexion sur l'apprentissage

47. Quel aspect de cet exercice avez-vous trouvé le plus difficile ? Le plus instructif ?