Sujets retenus

Les sujets suivants, au nombre de 14, sont d'une difficulté assez homogène. Ils couvrent de nombreuses notions du programme du cycle terminal en NSI et aucune bibliothèque n'est à installer. Nous proposons donc de retenir cette liste pour le choix de l'épreuve pratique de NSI le vendredi 19 Juin 2026.

Liste des sujets :

$$2 - 6 - 7 - 10 - 11 - 12 - 13 - 14 - 15 - 17 - 18 - 19 - 20 - 21$$

La plupart des sujets ayant été récemment modifiés, vous trouverez sur le lien, la dernière version des 14 sujets retenus parmi lesquels on retrouve l'un des sujets zéros. Nous attirons votre attention sur deux points : le sujet 13 qui nécessite par exemple que les élèves puissent avoir le droit de création/modification de fichiers comme des .kml. En ce sens, copier le dossier 26_BCG_NSI_13 sur le bureau est une pratique qui s'est avérée fonctionnelle. En outre, de nombreux sujets comme le 15 nécessitent de lire une donnée ; il faut donc bien veiller à ne pas ouvrir l'interpréteur avant d'ouvrir le fichier pour éviter d'éventuels soucis de chemin.

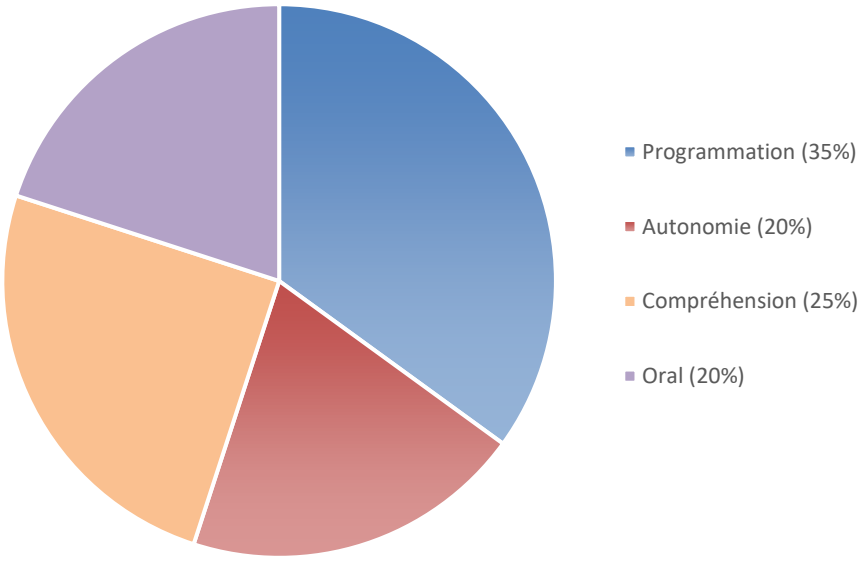
Nous vous rappelons qu'il faut donner des sujets différents pour chaque candidat d'une même session et qu'il faut effectuer des sessions différentes (sauf si mise en loge). N'hésitez pas à demander aux chefs d'établissements de vos lieux de passage une mise en loge par demi-journée.

Barème

La session 2026 des épreuves pratiques propose une notation sur 16 points, ramenée sur 20 par un algorithme.

Pour chacune des quatre compétences (Programmation / Autonomie / Compréhension / Oral), il faudra cocher un niveau (Insuffisant / Fragile / Suffisant / Maitrisé) et Santorin fournira la note finale à reporter sur la fiche individuelle d'aide à l'évaluation.

Au final les compétences compteront comme noté ci-dessous pour tous les sujets.



Questions aux candidats

Voici une liste de questions qui pourront vous aider à évaluer la partie orale de l'épreuve.

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 ?