CAPES NSI M2 – session 2026

Épreuve disciplinaire appliquée

Cette épreuve est constituée de trois parties : A, B et C indépendantes et de 9 annexes .
Les réponses aux questions doivent être précises et rédigées avec soin.

Partie A

On donne un extrait du programme de spécialité en classe de première NSI :

Contenus	Capacités attendues	Commentaires
Recherche dichotomique dans un tableau trié	Montrer la terminaison de la recherche dichotomique à l'aide d'un variant de boucle.	Des assertions peuvent être utilisées. La preuve de la correction peut être présentée par le professeur.

1. Vous souhaitez initier vos élèves de première de spécialité NSI à la recherche dichotomique, en utilisant une énigme du concours Castor présentée en annexe 1. Cette annexe 1 comprend également un exemple de résolution par un élève.
   Cependant, un problème technique vous empêche l'utilisation de cette activité.
   Proposez une autre activité, cette fois-ci débranchée (sans recours à un ordinateur), que vous pourriez proposer à vos élèves afin de leur faire découvrir la recherche dichotomique.
   Vous détaillerez l'activité en fournissant son nom, son lien avec la recherche dichotomique et en expliquant les grandes étapes lors de son déroulement.

2. Donnez et explicitez un exemple de recherche dichotomique dans un tableau trié, tel que vous pourriez l'écrire dans le cours de vos élèves de première. Votre présentation devra permettre aux élèves d'avoir un exemple clair et précis à inclure dans leur leçon.

Voici un exercice donné à des élèves de première spécialité NSI.

Écrire une fonction `recherche` qui prend en paramètres un tableau `tab` de nombres entiers triés par ordre croissant et un nombre entier `n`, et qui effectue une recherche dichotomique du nombre entier `n` dans le tableau non vide `tab`.

Cette fonction doit renvoyer un indice correspondant au nombre cherché s'il est dans le tableau, `None` sinon.

Exemples :

```
>>> recherche([2, 3, 4, 5, 6], 5)
3
>>> recherche([2, 3, 4, 6, 7], 5) # renvoie None
```

3. Proposez un corrigé de cet exercice.

4. Vous trouverez, dans l'annexe 2, un exercice conçu pour des élèves de première NSI dans le cadre d'une évaluation.
   Donnez les réponses correctes à cet exercice, sans les justifier.

5. Un élève vous demande d'expliciter les réponses des questions 2 et 3 de l'évaluation de l'annexe 2. Détaillez, pour chacune de ces deux questions, les réponses que vous pourriez lui apporter.
6. Démontrez la terminaison de l'algorithme de recherche dichotomique dans un tableau trié écrit à la question 3.

7. Démontrez la correction de l'algorithme de recherche dichotomique dans un tableau trié écrit à la question 3.

Dans la suite de cette partie, vous vous mettrez dans la position d'un professeur enseignant devant une classe de terminale spécialité NSI.

On donne un extrait du programme du programme de la spécialité en classe de terminale NSI :

Contenus	Capacités attendues	Commentaires
Récursivité	Écrire un programme récursif.	Analyser le fonctionnement d'un programme récursif. Des exemples relevant de domaines variés sont à privilégier.

8. Vous souhaitez réinvestir la recherche dichotomique dans un tableau trié, en lien avec le chapitre de la récursivité abordé en terminale. Rédigez un exercice demandant à vos élèves de terminale NSI de compléter un programme à « trous » implémentant la recherche dichotomique de manière récursive.
9. Proposez une correction pour l'exercice proposé à la question précédente.

Partie B :

On rappelle que la stéganographie est une technique consistant à dissimuler un message ou des données à l'intérieur d'un autre fichier (texte, image, audio, vidéo...).

Vous souhaitez faire travailler un groupe d'élèves de **première spécialité NSI** sur un projet abordant la stéganographie. Vous vous inspirez du projet disponible dans l'annexe 3 issue du « *vademecum Sciences numériques et technologie, Numérique et sciences informatiques* ».

10. En utilisant les extraits du programme de première NSI disponibles dans l'annexe 4, citez l'ensemble des contenus travaillés dans le cadre du programme de NSI à travers ce projet.

Dans l'annexe 5, vous trouverez un extrait du « *vademecum Sciences numériques et technologie, Numérique et sciences informatiques* » dans lequel sont proposés des critères d'évaluation lors des projets.

11. À partir de cet extrait, proposez une **grille d'évaluation de ce projet à destination de vos élèves** avec un barème sur 20 permettant de noter là la fois leur code source et leur compte rendu sur ce projet. Pensez à utiliser un **vocabulaire accessible pour vos élèves** et justifiez votre barème en quelques lignes en dessous de votre grille.

Voici un extrait du « *vademecum Sciences numériques et technologie, Numérique et sciences informatiques* », concernant la place de l'oral dans l'enseignement de la NSI.

> *L'épreuve du Grand oral vise avant tout à mobiliser et évaluer les compétences oratoires des élèves. Il est indispensable que ces compétences soient travaillées au lycée, et ce, dès la classe de seconde en continuité des apprentissages du collège.*
> *[…]*
> *L'oral est à la fois un objet à apprendre et un outil pour apprendre. Les enseignants doivent expliciter aux élèves les compétences orales attendues et les mobiliser lors de situations variées. L'explicitation des compétences orales peut être facilitée par l'utilisation avec les élèves de la grille d'évaluation indicative de l'épreuve orale terminale.*

Par ailleurs, en annexe 6, la grille d'évaluation de l'épreuve du grand oral, vous est donnée.

12. En vous inspirant des critères de cette grille, et en l'ajustant au fait que le passage à l'oral de votre projet s'effectuera en groupe, proposez une **grille d'évaluation explicite à destination de vos élèves** afin de préparer leur passage à l'oral sur ce projet sur 10 points.

Dans la suite de cette partie, vous vous mettrez dans la position d'un professeur enseignant devant une classe de terminale spécialité NSI.

Vous abordez la **sécurisation de communication** dont voici un extrait du programme de Terminale spécialité NSI.

Contenus	Capacités attendues	Commentaires
Sécurisation des communications	Décrire les principes de chiffrement symétrique (clef partagée) et asymétrique (avec clef privée/clef publique). Décrire l'échange d'une clef symétrique en utilisant un protocole asymétrique pour sécuriser une communication HTTPS.	Les protocoles symétriques et asymétriques peuvent être illustrés en mode débranché, éventuellement avec description d'un chiffrement particulier. La négociation de la méthode chiffrement du protocole SSL (*Secure Sockets Layer*) n'est pas abordée

13. Un élève vous dit : « l'année dernière on a déjà vu un chiffrement symétrique lors du projet sur la stéganographie ». Donnez une explication claire et succincte que vous pourriez lui apporter sur son erreur.

Dans le programme de Terminale spécialité NSI, voici un extrait relevant de l'histoire de l'informatique.

Contenus	Capacités attendues	Commentaires
Événements clés de l'histoire de l'informatique	Situer dans le temps les principaux événements de l'histoire de l'informatique et leurs protagonistes. Identifier l'évolution des rôles relatifs des logiciels et des matériels.	Ces repères viennent compléter ceux qui ont été introduits en première. Ces repères historiques sont construits au fur et à mesure de la présentation des concepts et techniques.

14. Citez quatre repères historiques importants permettant de montrer l'évolution de la sécurisation des données au cours de l'histoire, que vous pourriez intégrer à votre leçon sur la **sécurisation de communication**

15. Proposez le déroulé d'une activité d'informatique débranchée permettant d'introduire le chiffrement symétrique devant une classe. Vous donnerez une explication détaillée, en une dizaine de lignes, des points que vous estimez importants ou délicats à comprendre pour les élèves.

16. Expliquez devant une classe comment vous mettriez en évidence les limites du chiffrement symétrique.

17. Donnez deux schémas récapitulatifs que vous pouvez présenter aux élèves afin de présenter les chiffrements symétrique et asymétrique. Prenez soin de mettre en dessous de chacun d'eux une phrase expliquant chacun d'entre eux.

Partie C :

Dans cette partie on propose d'étudier les réseaux sociaux et des problématiques algorithmiques sous-jacentes au travers des programmes de SNT, première NSI et terminale.

o C-1 - Réseaux sociaux

Dans un premier temps vous souhaitez aborder avec des élèves de seconde en SNT des problématiques liées aux réseaux.

18. Définissez le concept d'**identité numérique** tel que vous le feriez devant des élèves.
19. Donnez deux exemples d'usages qui peuvent être fait avec les informations personnelles qu'ils partagent sur les réseaux sociaux.
20. Quel « conseil pratique » leur donneriez-vous pour protéger leurs données personnelles ?
21. Les réseaux sociaux sont gratuits pour les utilisateurs, mais pas pour les entreprises qui les mettent en œuvre. Donnez deux manières dont ces entreprises parviennent à gagner de l'argent. Vos explications devront être compréhensibles par des élèves de seconde.

Le programme de SNT mentionne l'expérience du "petit monde" de Milgram pour évoquer les problématiques liées aux informations qu'on trouve sur les réseaux sociaux.

Contenus	Capacités attendues
Notion de « petit monde » Expérience de Milgram	Décrire comment l'information présentée par les réseaux sociaux est conditionnée par le choix préalable de ses amis.

22. Expliquer le principe du « petit-monde » de Milgram comme vous l'expliqueriez à vos élèves de seconde.

23. Proposer une activité sans ordinateur pour illustrer le principe du petit monde dans le cadre de votre classe. On demande une description de l'activité et de son matériel.
    On ne dépassera pas 15 lignes pour l'explication. Des schémas peuvent être ajoutés si nécessaires.

o C-2 - Amis

Dans cette partie, vous enseignez l'algorithme des k plus proches voisins à vos élèves de Première NSI.
Les attendus du programme sont les suivants :

Contenus	Capacités attendues	Commentaires
Algorithme des k plus proches voisins	Écrire un algorithme qui prédit la classe d'un élément en fonction de la classe majoritaire de ses k plus proches voisins.	Il s'agit d'un exemple d'algorithme d'apprentissage

Vous voulez motiver l'utilité de cet algorithme par le biais des algorithmes de recommandation d'amis sur les réseaux sociaux.

24. Expliquer le principe de l'algorithme des k plus proches voisins tel que vous l'expliqueriez à des élèves de première.

25. Donnez deux exemples de distances pouvant être utilisées.

26. Sur un exemple de classification de points du plan en 3 classes, montrer avec 3 dessins l'importance du choix de la variable k.

L'annexe 7 propose un énoncé d'exercice sur l'algorithme des k plus proches voisins.

27. Proposez une correction de cet exercice.

o C-3 - Graphes

Dans cette partie, vous souhaitez illustrer les réseaux sociaux avec des élèves de seconde en SNT. Le programme de SNT demande de présenter des notions de graphes.

Contenus	Capacités attendues
Rayon, diamètre et centre d'un graphe	Déterminer ces caractéristiques sur des graphes simples.

28. Citer deux exemples, autre que les réseaux sociaux, montrant que les graphes permettent de modéliser la réalité.

29. Expliquer la notion de rayon, diamètre et centre d'un graphe et faire le lien avec le phénomène du petit monde.

30. Rédiger un exercice illustrant les 3 notions précédentes, comportant 4 questions, pour vos élèves de seconde.

Dans la suite de cet exercice, vous vous mettrez dans la position d'un professeur enseignant devant une classe de terminale spécialité NSI.

Le programme de Terminale solidifie ces notions afin de préparer les étudiants aux parcours et à la programmation.

Contenus	Capacités attendues	Commentaires
Graphes : structures relationnelles. Sommets, arcs, arêtes, graphes orientés ou non orientés.	Modéliser des situations sous forme de graphes. Écrire les implémentations correspondantes d'un graphe : matrice d'adjacence, liste de successeurs/de prédécesseurs. Passer d'une représentation à une autre.	On s'appuie sur des exemples comme le réseau routier, le réseau électrique, Internet, les réseaux sociaux. Le choix de la représentation dépend du traitement qu'on veut mettre en place : on fait le lien avec la rubrique « algorithmique ».
Algorithmes sur les graphes.	Parcourir un graphe en profondeur d'abord, en largeur d'abord. Repérer la présence d'un cycle dans un graphe. Chercher un chemin dans un graphe.	Le parcours d'un labyrinthe et le routage dans Internet sont des exemples d'algorithme sur les graphes. L'exemple des graphes permet d'illustrer l'utilisation des classes en programmation.

31. Quelle définition des graphes allez vous donner à vos élèves de Terminale?

32. Vous avez réalisé un cours complet sur les graphes et leurs représentations.
Décrivez maintenant un plan de cours (décrivant notions et activités abordées) sur les parcours de graphes, séparé en séances, en 15 lignes.

33. Un élève vous demande l'intérêt d'avoir deux représentations possibles pour les graphes, en liste d'adjacence ou en matrice d'adjacence. Que répondez-vous?

34. Vous avez donné l'exercice en Annexe 8 à vos élèves. Proposez un barème sur 10 pour cet exercice, en le justifiant. Puis, corrigez la copie d'élève en annexe 9 selon ce barème.

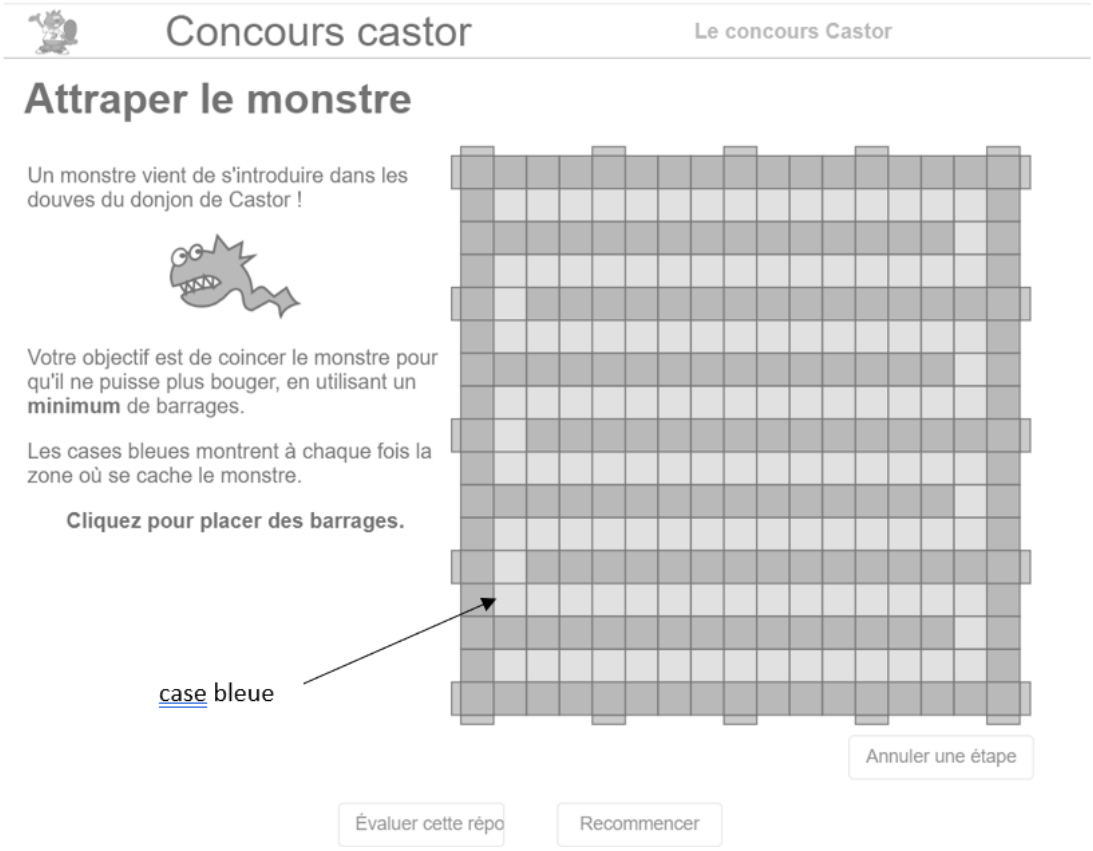
35. Donner la complexité de la méthode `defiler` de la classe `File` présentée dans l'annexe 8. Comment pourriez-vous expliquer simplement ce coût à vos élèves de Terminale afin qu'ils en comprennent l'origine ?

36. Proposez une autre implémentation de la classe `File` qui permette d'obtenir une méthode `defiler` plus efficace, avec un coût inférieur à celui de l'implémentation actuelle. Votre proposition ne devra pas utiliser de bibliothèques supplémentaires.

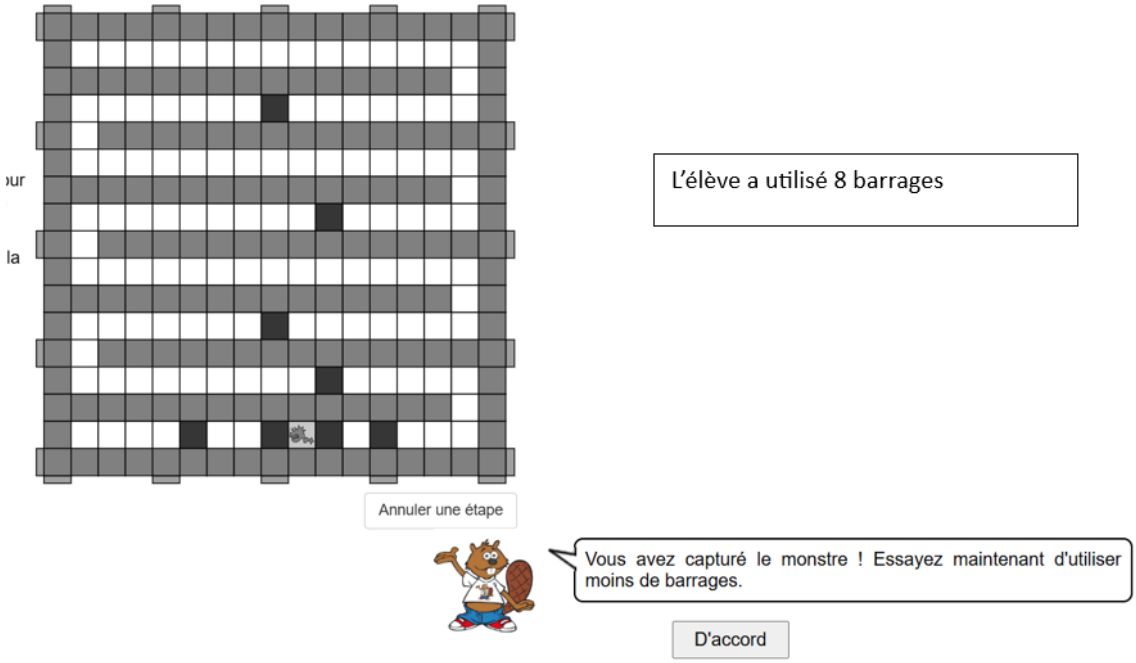
Annexes :

Annexe 1 :

Annexe 1-A : capture d'écran de l'énigme à résoudre



Annexe 1-B : capture d'écran d'un élève



Annexe 2 :

Exercice proposé à des élèves de première de spécialité NSI lors d'une évaluation

Exercice n° 1 — 2 points

Cet exercice est un QCM. Pour chacune de ces questions, une seule des quatre réponses est exacte. Coche la réponse exacte. Aucune justification n'est demandée.

Question 1 :
La recherche dichotomique est un algorithme rapide qui permet de trouver ou non la présence d'un élément dans un tableau. Mais, pour l'utiliser, une contrainte est indispensable, laquelle ?

- ☐ le tableau ne contient que des nombres positifs
- ☐ la longueur du tableau est une puissance de 2
- ☐ le tableau est trié en ordre croissant
- ☐ le tableau ne contient pas la valeur 0

Question 2 :
Quelle valeur permet de compléter l'affirmation suivante : « Le nombre d'opérations nécessaires pour rechercher un élément séquentiellement dans un tableau de longueur n est de l'ordre de ... »

☐ 1 ☐ n ☐ n^2 ☐ n^3

Question 3 :
En utilisant une recherche dichotomique, combien faut-il de comparaisons avec l'opérateur == pour trouver une valeur dans un tableau trié de 1000 nombres, dans le pire des cas ?

☐ 3 ☐ 10 ☐ 1000 ☐ 1024

Question 4 :
Un algorithme de recherche dichotomique dans une liste triée de taille n nécessite, dans le pire des cas, exactement k comparaisons. Combien cet algorithme va-t-il utiliser, dans le pire des cas, de comparaisons sur une liste de taille $2n$?

☐ $2k$ ☐ $k+1$ ☐ k ☐ $2k+1$

Annexe 3 :

Extrait du vadémécum Sciences numériques et technologie, Numérique et sciences informatiques

L'exemple proposé est un projet de programmation en Python accessible aux élèves en classe de première NSI. Il consiste, à partir d'un travail sur des images, à mixer une image support et une image message sans que celle-ci se remarque dans le résultat du mixage.

Dans le codage Rouge-Vert-Bleu (RVB), chaque pixel est la combinaison d'une déclinaison des 3 couleurs primaires et chaque couleur est codée sur 8 bits, soit 24 bits au total par pixel. Chaque bit prenant la valeur 0 ou 1 (2 possibilités), cela fait 2^{24} = 16 777 216 couleurs différentes possibles par pixel. C'est beaucoup plus que l'œil humain ne peut en distinguer.

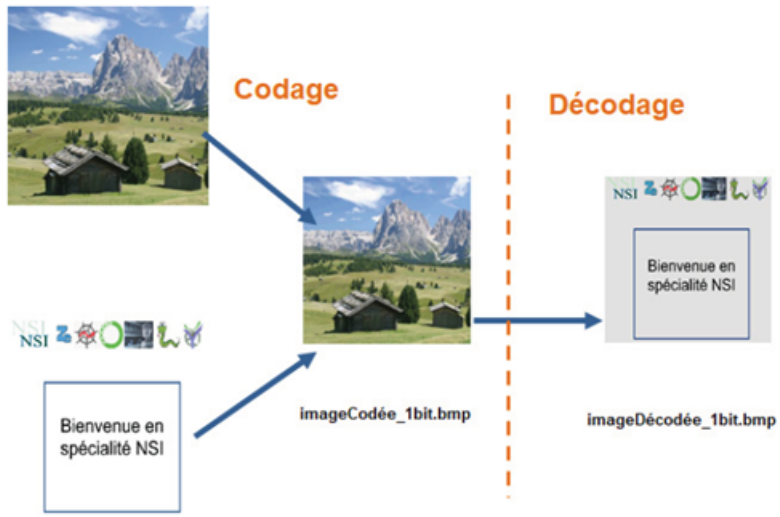
En pratique, si une couleur est stockée sur l'octet [s7, s6, s5, s4, s3, s2, s1, s0], l'information prioritaire se situe au niveau des bits de poids forts. On suppose que les deux images sont de même taille.

Si on fixe n=3, si l'octet de l'image support est

[**s7**, **s6**, **s5**, **s4**, **s3**, s2, s1, s0],
Si l'octet de l'image message est
[**m7**, **m6**, **m5**, m4, m3, m2, m1, m0],
Alors l'octet de l'image mixée est
[**s7**, **s6**, **s5**, **s4**, **s3**, **m7**, **m6**, **m5**].
Pour reconstruire l'image cachée, on reconstruit les octets
[**m7**, **m6**, **m5**, 0, 0, 0, 0, 0].

L'image cachée est un peu altérée, mais c'est largement suffisant si le message caché est simple.

Exemple :



Annexe 4 :

Extraits du programme du programme de spécialité en classe de première NSI

[…]

Représentation des données : types et valeurs de base

Contenus	Capacités attendues	Commentaires
Écriture d'un entier positif dans une base $b \geqslant 2$	Passer de la représentation d'une base dans une autre.	Les bases 2, 10 et 16 sont privilégiées.
Représentation binaire d'un entier relatif	Évaluer le nombre de bits nécessaires à l'écriture en base 2 d'un entier, de la somme ou du produit de deux nombres entiers. Utiliser le complément à 2.	Il s'agit de décrire les tailles courantes des entiers (8, 16, 32 ou 64 bits). Il est possible d'évoquer la représentation des entiers de taille arbitraire de Python.
Représentation approximative des nombres réels : notion de nombre flottant	Calculer sur quelques exemples la représentation de nombres réels : 0.1, 0.25 ou 1/3.	0.2 + 0.1 n'est pas égal à 0.3. Il faut éviter de tester l'égalité de deux flottants. Aucune connaissance précise de la norme IEEE-754 n'est exigible.
Valeurs booléennes : 0, 1. Opérateurs booléens : and, or, not. Expressions booléennes	Dresser la table d'une expression booléenne.	Le ou exclusif (xor) est évoqué. Quelques applications directes comme l'addition binaire sont présentées. L'attention des élèves est attirée sur le caractère séquentiel de certains opérateurs booléens.
Représentation d'un texte en machine. Exemples des encodages ASCII, ISO-8859-1, Unicode	Identifier l'intérêt des différents systèmes d'encodage. Convertir un fichier texte dans différents formats d'encodage.	Aucune connaissance précise des normes d'encodage n'est exigible.

[…]

Représentation des données : types construits

Contenus	Capacités attendues	Commentaires
p-uplets. p-uplets nommés	Écrire une fonction renvoyant un p-uplet de valeurs.	
Tableau indexé,	Lire et modifier les éléments d'un	Seuls les tableaux dont les éléments

tableau donné en compréhension	tableau grâce à leurs index. Construire un tableau par compréhension. Utiliser des tableaux de tableaux pour représenter des matrices : notation a [i] [j]. Itérer sur les éléments d'un tableau.	sont du même type sont présentés. Aucune connaissance des tranches (*slices*) n'est exigible. L'aspect dynamique des tableaux de Python n'est pas évoqué. Python identifie listes et tableaux. Il n'est pas fait référence aux tableaux de la bibliothèque NumPy.
Dictionnaires par clés et valeurs	Construire une entrée de dictionnaire. Itérer sur les éléments d'un dictionnaire.	Il est possible de présenter les données EXIF d'une image sous la forme d'un enregistrement. En Python, les p-uplets nommés sont implémentés par des dictionnaires. Utiliser les méthodes *keys*(), *values* () et *items* ().

[…]

Langages et programmation

Contenus	Capacités attendues	Commentaires
Constructions élémentaires	Mettre en évidence un corpus de constructions élémentaires.	Séquences, affectation, conditionnelles, boucles bornées, boucles non bornées, appels de fonction.
Diversité et unité des langages de programmation	Repérer, dans un nouveau langage de programmation, les traits communs et les traits particuliers à ce langage.	Les manières dont un même programme simple s'écrit dans différents langages sont comparées.
Spécification	Prototyper une fonction. Décrire les préconditions sur les arguments. Décrire des postconditions sur les résultats.	Des assertions peuvent être utilisées pour garantir des préconditions ou des postconditions.
Mise au point de programmes	Utiliser des jeux de tests.	L'importance de la qualité et du nombre des tests est mise en évidence. Le succès d'un jeu de tests ne garantit pas la correction d'un programme.

Annexe 5 :

Extrait du vadémécum Sciences numériques et technologie, Numérique et sciences informatiques

[…]

Un mini-projet dure moins de 8 heures et est réalisé individuellement ou en binôme. Il est possible d'évaluer les acquis des élèves à partir d'un compte rendu individuel, du code source et d'un oral. Les critères d'évaluation du compte rendu pourraient être :

- la pertinence de la répartition des tâches ;
- l'analyse des problèmes à résoudre ;
- la clarté de l'expression et des illustrations ;
- l'explicitation des méthodes et des choix effectués ;
- l'identification et le réinvestissement de notions vues en cours ;
- l'analyse critique des solutions.

Les critères d'évaluation du code source pourraient être :

- le respect du cahier des charges ;
- pour la partie programmation :
  - la pertinence de la décomposition de la tâche sous forme de sous-tâches (fonctions, modules, bibliothèques, variables, structures de données, etc.) ;
  - la facilité de lecture du code : indentation, existence d'une documentation, de commentaires, de noms explicites pour les variables ;
  - l'interopérabilité des programmes réalisés par chaque élève ;
  - la prise en compte de la sécurité numérique quand cela est nécessaire ;
- pour la partie interface graphique :
  - l'ergonomie du produit final (clarté, intuitivité, contenu, etc.).

Annexe 6 :

Extrait de la note de service relative aux modalités d'évaluation des candidats à l'épreuve du grand oral (version consolidé juin 2024)

Annexe 1 - Grille d'évaluation indicative de l'épreuve orale terminale

	Qualité orale de l'épreuve	Qualité de la prise de parole en continu	Qualité des connaissances	Qualité de l'interaction	Qualité de construction de l'argumentation
Très insuffisant	Difficilement audible sur l'ensemble de la prestation. Le candidat ne parvient pas à capter l'attention.	Énoncés courts, ponctués de pause et de faux démarrages ou énoncés longs à la syntaxe mal maîtrisée.	Connaissances imprécises, incapacité à répondre aux questions, même avec une aide et des relances.	Réponses courtes ou rares. La communication repose principalement sur l'évaluateur.	Pas de compréhension du sujet, discours non argumenté et décousu.
Insuffisant	La voix devient plus audible et intelligible au fil de l'épreuve mais demeure monocorde. Vocabulaire limité ou approximatif	Discours assez clair mais vocabulaire limité et énoncés schématiques.	Connaissances réelles, mais difficultés à les mobiliser en situation à l'occasion des questions du jury.	L'entretien permet une amorce d'échange. L'interaction reste limitée.	Début de démonstration mais raisonnement lacunaire. Discours insuffisamment structuré.
Satisfaisant	Quelques variations dans l'utilisation de la voix ; prise de parole affirmée. Il utilise un lexique adapté. Le candidat parvient à susciter l'intérêt.	Discours articulé et pertinent, énoncés bien construits.	Connaissances précises, une capacité à les mobiliser en réponses aux questions du jury avec éventuellement quelques relances.	Répond, contribue, réagit. Se reprend, reformule en s'aidant des propositions du jury.	Démonstration construite et appuyée sur des arguments précis et pertinents.
Très satisfaisant	La voix soutient efficacement le discours. Qualités prosodiques marquées (débit, fluidité, variations et nuances pertinentes, etc.). Le candidat est pleinement engagé dans sa parole. Il utilise un vocabulaire riche et précis.	Discours fluide, efficace, tirant pleinement profit du temps et développant des propositions.	Connaissances maîtrisées, les réponses aux questions du jury témoignent d'une capacité à mobiliser ces connaissances à bon escient et à les exposer clairement.	S'engage dans sa parole, réagit de façon pertinente. Prend l'initiative dans l'échange. Exploite judicieusement les éléments fournis par la situation d'interaction.	Maîtrise des enjeux du sujet, capacité à conduire et exprimer une argumentation personnelle, bien construite et raisonnée.

Annexe 7 : Exercice sur les k plus proches voisins :

On considère un réseau social où des personnes se regroupent et échangent selon leurs hobbys : la cuisine, le jardinage, le bricolage ou la lecture. Chaque utilisateur peut publier des images ou des messages en rapport avec ses hobbys et aimer le contenu des autres.

Pour proposer à chaque utilisateur de nouveaux amis partageant ses passions, le réseau social note l'intérêt de chaque utilisateur pour chaque hobby par une note entre 0 et 10, en fonction de ses publications.

Pour chaque personne, les quatre notes sont regroupées dans une liste. Par exemple, la liste de Jeanne est **[0, 2, 9, 10]** car Jeanne n'aime pas la cuisine, n'aime pas trop le jardinage, aime beaucoup le bricolage et encore plus la lecture. Les hobbys sont toujours rangés dans l'ordre qu'on vient d'énumérer.

Voici la liste des utilisateurs, la liste de leur hobbys et si ce sont des amis de Jeanne :

Nom	**Hobbys**	**Ami ?**
Edgar	**[1, 1, 8, 8]**	oui
Maélys	**[9, 9, 0, 0]**	non
Camille	**[10, 2, 5, 5]**	non
Évelyne	**[3, 10, 10, 3]**	oui
Coralie	**[8, 0, 0, 9]**	oui
Jean	**[9, 2, 2, 4]**	non
Émile	**[4, 4, 6, 8]**	oui

Louis, un nouvel utilisateur, a comme tableau **[10, 0, 3, 5]**.

1. Ranger les utilisateurs selon la similarité de leurs intérêts avec ceux de Louis. On utilisera la distance euclidienne pour caractériser la similarité des intérêts.

2. Écrire une fonction **`distance`** qui calcule la distance euclidienne entre deux listes de préférences. On pourra supposer le module math importé.
   Par exemple **`distance([1,1,8,8],[10,0,3,5]) = 10.770329614269007`**

3. Le réseau devrait-il proposer Louis comme ami à Jeanne si on considère les k=2 utilisateurs ayant les intérêts les plus similaires à Louis? Les k=5 plus similaires?

4. On propose le code suivant incomplet d'une fonction `classifie` pour classifier si un nouvel arrivant **`x`** devrait être ami de Jeanne, en fonction de ses goûts, selon le principe de l'algorithme des k plus proches voisins vu en cours.

Les paramètres de cette fonction sont les suivantes :

- Le dictionnaire **`hobbys`** où sont stockés les goûts des utilisateurs,

Par exemple **`hobbys["Edgar"] = [1,1,8,8]`**.

- Le dictionnaire **`amis`** indique si un utilisateur est ami de Jeanne.
  Par exemple **`amis["Coralie"] = False`**.
- **`goutsx`** est la liste des gouts du nouvel arrivant **`x`**.
- **`k`** est le paramètre de l'algorithme des k plus proches voisins.

La sortie de cette fonction est **`True`** si on classifie **`x`** comme un ami, **`False`** sinon.

```
1 def classifie(hobbys, amis, goutsx, k) :
2     """ Détermine si un nouvel utilisateur doit être classifié comme
3     un ami ou non """
4
5     #La liste des distances contiendra la distance et la classe,
6     #par exemple (5, True) ou (12.23353, False)
7     liste_distances = []
8     #Pour chaque utilisateur on calcule sa distance au nouveau venu
9     for utilisateur in ... :
10         liste_distances.append(...)
11
12    #Réordonner la liste pour mettre les voisins les plus proches
13    #au début
14    …  # code sur plusieurs lignes possible
15
16    #Parmi les k plus proches voisins,compter les amis et les non-amis
17    nbVrai = 0
18    nbFaux = 0
19    for i in range(k) :
20        dist, booleen = ...
21        if booleen :
22            ...
23        else :
24            ...
25
26    #On conclut sur la classe majoritaire
27    if  …  :
28        return True
29    else :
30        return False
```

Recopier et compléter les lignes comportant des trous de cette fonction.

Annexe 8 : Exercice sur les graphes :

Dans cet exercice, on considère que les sommets des graphes sont numérotés par des entiers entre 0 et n-1, où n est le nombre de sommets du graphe.

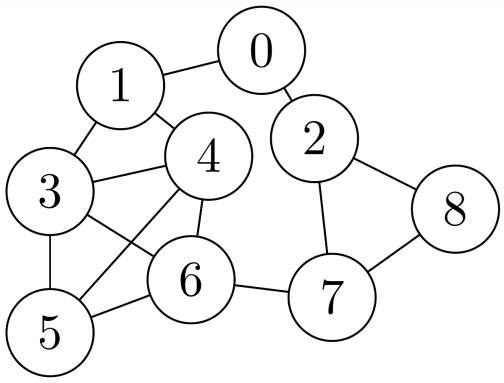
1. Dessiner le graphe correspondant à la liste d'adjacence suivante :
   `L = [[1], [0, 2, 4], [1, 3, 4], [2, 4], [1, 2, 3]]`.
2. Dessiner le graphe correspondant à la matrice d'adjacence A. Ce graphe est-il orienté ou non orienté? On justifiera la réponse.

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

3. Écrire une fonction **`degres`** qui prend en entrée un graphe représenté par sa matrice d'adjacence et renvoie une liste L telle que L[i] est le nombre de voisins du sommet i.

   Par exemple **`degres([[0,1,0,0], [1,0,1,1], [0,1,0,1], [0,1,1,0]])`** renvoie **`[1, 3, 2, 2]`**.

On considère le graphe non-orienté suivant :



4. Donner l'ordre dans lequel les sommets sont parcourus par un parcours en profondeur à partir de s=0 et par un parcours en largeur à partir de s=0.
   À chaque étape on considérera les voisins dans l'ordre croissant de leur numéros.

On suppose une file implémentée avec l'interface suivante :

```
class File:
    """ classe file """
    def __init__(self):
        self.lst = []
    def __repr__(self):
        return str(self.lst)
    def est_vide(self):
        return len(self.lst) == 0
    def enfiler(self, v):
        self.lst.append(v)
    def defiler(self):
        return self.lst.pop(0)
```

5. Écrire en Python une fonction **`parcours_largeur`** qui effectue le parcours en largeur sur un graphe représenté par sa matrice d'adjacence et affiche les numéros des sommets dans l'ordre de parcours, séparés par des virgules.

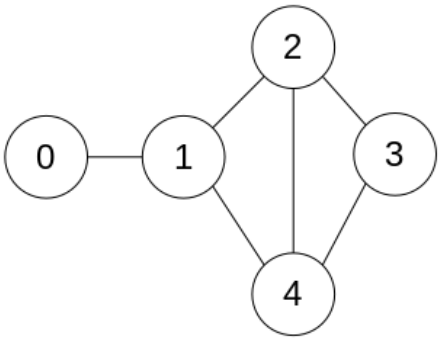
La fonction prendra en entrée la matrice d'adjacence du graphe **`mat_adj`** et le sommet de départ du parcours **`s`**.

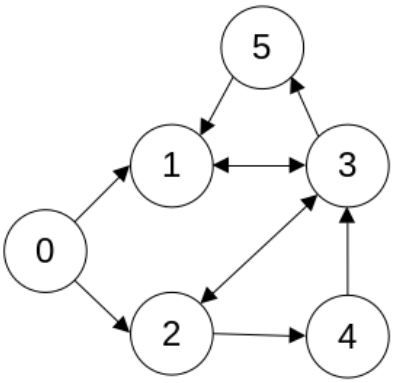
Exemples :

`parcours_largeur([[0, 1], [1, 0]], 0)` affiche **`0,1`**

`parcours_largeur([[0, 1, 1, 0], [1, 0, 0, 0], [1, 0, 0, 1], [0, 0, 0, 1]], 0)` affiche **`0, 1, 2, 3`**

Annexe 9 : Copie d'élève :

1. 

2. 

Le graphe est orienté.

3.
```
def degres(mat_adj) :
    n = len(mat_adj)
    L = [0] * n
    for i in range(n) :
        L[i] = len(mat_adj[i])
    return L
```

4. Pour le parcours en profondeur : 0, 1, 3, 4, 5, 6, 7, 2, 8
   Pour le parcours en largeur : 0, 1, 2, 3, 4, 7, 8, 5, 6

5.
```
def parcours_largeur(mat_adj, s) :
    f = File
    f.enfiler(s)
    while not est_vide(f) :
        x = f.defiler()
        print(x,", ")
        for i in range(len(mat_adj)) :
            if mat_adj[x][i] = 1 :
                f.enfiler(i)
```