

Préparation Epreuves / Projet

formation NSI

version 24

Journée Préparation aux Epreuves de NSI

Qui sommes-nous ?

- ▶ Véronique Glaçon : veronique-Miche.dewatine@ac-lille.fr
- ▶ Hervé Owsinski : herve.owsinski@ac-lille.fr

1 - Déroulé

Matin : Préparations aux épreuves du BAC

1 - Déroulé

Matin : Préparations aux épreuves du BAC

Après-midi : les projets

1 - Déroulé

Matin : Préparations aux épreuves du BAC

Après-midi : les projets

2 - Pour commencer : Le vademecum

<https://pedagogie.ac-lille.fr/numerique-et-sciences-informatiques/nouveau-document-pedagogique-le-vademecum-snt-nsi/>



VADEMECUM



2 - Pour commencer : Le vademecum



- La différenciation pédagogique : en adaptant le niveau de difficulté, la quantité de travail à réaliser, ou encore l'autonomie laissée aux élèves, et en utilisant certains outils numériques.

2 - Pour commencer : Le vademecum



- ▶ La différenciation pédagogique : en adaptant le niveau de difficulté, la quantité de travail à réaliser, ou encore l'autonomie laissée aux élèves, et en utilisant certains outils numériques.
- ▶ L'enseignement de NSI doit être dynamique, mettant en oeuvre les démarches d'investigation, de résolution de problème et de projet.

2 - Pour commencer : Le vademecum



- ▶ La différenciation pédagogique : en adaptant le niveau de difficulté, la quantité de travail à réaliser, ou encore l'autonomie laissée aux élèves, et en utilisant certains outils numériques.
- ▶ L'enseignement de NSI doit être dynamique, mettant en oeuvre les démarches d'investigation, de résolution de problème et de projet.
- ▶ L'utilisation individuelle d'un ordinateur est indispensable durant au moins la moitié du temps en classe.

2 - Pour commencer : Le vademecum



- proposer les différents éléments de cours sur l'ENT. Afin d'être réutilisables et lisibles lors des révisions, les programmes doivent comporter aussi des commentaires.

2 - Pour commencer : Le vademecum



- ▶ proposer les différents éléments de cours sur l'ENT. Afin d'être réutilisables et lisibles lors des révisions, les programmes doivent comporter aussi des commentaires.
- ▶ Il paraît donc indispensable que l'étude des algorithmes et la programmation soient réparties de façon spiralaire.

2 - Pour commencer : Le vademecum



- ▶ proposer les différents éléments de cours sur l'ENT. Afin d'être réutilisables et lisibles lors des révisions, les programmes doivent comporter aussi des commentaires.
- ▶ Il paraît donc indispensable que l'étude des algorithmes et la programmation soient réparties de façon spiralaire.
- ▶ La démarche de projet doit être mise en oeuvre régulièrement : avec une revue de projet.

2 - Pour commencer : Le vademecum



- ▶ Promouvoir les enseignements auprès des collégiens et des secondes, leurs enseignants et leur famille.
- ▶ Favoriser l'engagement des filles

3 - Sur l'évaluation

Le guide de l'évaluation préconise « qu'en fonction des objectifs poursuivis et selon les compétences évaluées, l'évaluation pourra s'appuyer sur des devoirs surveillés sur table ou sur machine, des devoirs en temps libre, des interrogations « flash », des présentations orales et des projets informatiques. »

Les devoirs sur table (formats longs) permettent de préparer les élèves à l'épreuve écrite finale du baccalauréat, mais sont aussi l'occasion de confronter les élèves à des résolutions de problèmes et des réflexions complexes.

Concernant l'évaluation sur machine, elle peut se faire : - sur des temps courts à l'aide de documents computationnels (notebook « Jupyter » ou Capytale) pour évaluer des connaissances et la compréhension et l'application directe du cours ; - sur des temps relativement courts avec un format proche des exercices proposés à l'épreuve pratique du bac pour évaluer les compétences du programme sur des problèmes simples ;

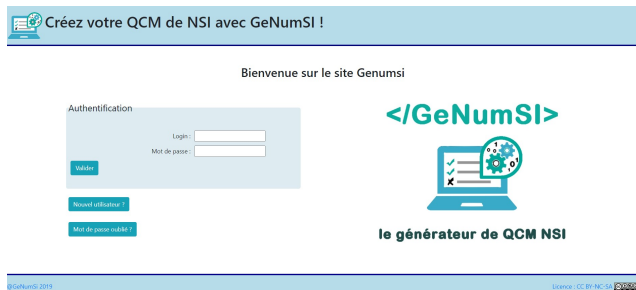
- ▶ sur des temps longs lors des mini-projets et des projets pour évaluer d'une part différentes compétences du programme au travers de problèmes complexes, et d'autre part des compétences liées à la gestion de projet (planifier, organiser, communiquer, travailler en équipe).

4 - Préparation aux épreuves

- ▶ Mise en place de qcm(s) de révisions ou de remédiation
- ▶ e-nsi et Codex
- ▶ Une remarque : apps.education.fr
- ▶ Préparations aux épreuves pratiques
- ▶ Utiliser un GIT ?
- ▶ Mise en place de rituels au début du cours

4.1 - Mise en place de qcm(s) de révisions ou de remédiation :

Par exemple, le site genumsi.inria.fr :



The screenshot shows the homepage of the GeNumSI website. At the top, a light blue banner contains the text "Créez votre QCM de NSI avec GeNumSI !" next to a small icon of a laptop with a gear. Below the banner, the text "Bienvenue sur le site Genumsi" is centered. On the left, there is an "Authentication" section with a light blue background, containing "Login:" and "Mot de passe:" labels, input fields, and a "Valider" button. Below this are two links: "Nouvel utilisateur ?" and "Mot de passe oublié ?". On the right, the text "</GeNumSI>" is displayed in a large, stylized font. Below it is an icon of a laptop with a gear and checkmarks, and the text "le générateur de QCM NSI". At the bottom, a light blue footer bar contains the text "©GeNumSI 2019" on the left and "Licence : CC BY-NC-SA" with a Creative Commons license icon on the right.

Voici deux liens vers des qcms générés aléatoirement:

- ▶ qcm première
- ▶ qcm terminale

4.2 - Préparation aux épreuves : e-nsi et Codex

Pas de préparation forte aux épreuves par les enseignants. . .

Comment faire ?

Une solution : permettre aux élèves de travailler en autonomie sur des sites spécialement créés pour cela :

- ▶ e-nsi dont l'adresse va changer sous peu puisque la forge de l'AEIF est destinée à disparaître, remplacée par la forge des communs du numérique.
- ▶ Codex, un fork du précédent.

Ces deux sites permettent d'entraîner les élèves en respectant leur autonomie.

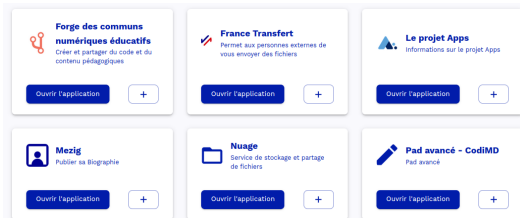
Dans les deux cas : CC BY-NC-SA 4.0

Une remarque : apps.education.fr

Le portail apps.education.fr fournit des outils adaptés à la RGPD.

Notamment :

- ▶ La Forge des Communs du Numérique → pour stocker vos sites, notamment en MkDocs.
- ▶ CodiMD → mais c'est plus pour communiquer entre professionnels.



4.3 - Préparations aux epreuves pratiques

Le vademecum :

Programmer, c'est résoudre un problème en construisant un programme qui pourra être compris, partagé et amélioré.

Pour cela, une attention particulière doit être portée pour rendre le code :

- ▶ facile à lire, en travaillant l'indentation et le choix des noms des variables et structures de données ;

4.3 - Préparations aux epreuves pratiques

Le vademecum :

Programmer, c'est résoudre un problème en construisant un programme qui pourra être compris, partagé et amélioré.

Pour cela, une attention particulière doit être portée pour rendre le code :

- ▶ facile à lire, en travaillant l'indentation et le choix des noms des variables et structures de données ;
- ▶ explicite, en indiquant des commentaires ;

4.3 - Préparations aux epreuves pratiques

Le vademecum :

Programmer, c'est résoudre un problème en construisant un programme qui pourra être compris, partagé et amélioré.

Pour cela, une attention particulière doit être portée pour rendre le code :

- ▶ facile à lire, en travaillant l'indentation et le choix des noms des variables et structures de données ;
- ▶ explicite, en indiquant des commentaires ;
- ▶ soigné, en éliminant les parties de programmes inutiles.

4.4 - Utiliser un GIT ?

Le principe est simple :

- ▶ répartir tous les sujets entre les élèves.
- ▶ leur demander de les faire chez eux MAIS de **commenter correctement chaque ligne** de code pour l'exercice 1 et les lignes importantes de l'exercice 2.
- ▶ on en choisit quelques uns et on tente de les critiquer pour améliorer ce que les élèves doivent être capables de faire le jour de l'épreuve.

4.4 - Utiliser un GIT ?

Exemple Epreuve 24, code d'élève :

```
def parcours_largeur(AB: 'Arbre Binaire') -> list:
    """Renvoie la liste des étiquettes rencontrées"""
    file = [AB]
    t = []
    while len(file) != 0:
        x = file.pop(0)
        t.append(x[1])
        if x[0] != None:
            file.append(x[0])
        if x[2] != None:
            file.append(x[2])
    return t
```


4.4 - Utiliser un GIT ?

Les commentaires de l'élève qui corrige permet d'aider les autres élèves... après correction du prof : leurs commentaires permet souvent de se rendre compte qu'ils ne comprennent pas exactement la notion.

```
def parcours_largeur(AB: 'Arbre Binaire') -> list:
    """Renvoie la liste des étiquettes rencontrées"""
    file = [AB] # on dit que [AB] est une file
    t = [] # on dit que t est un tableau
    while len(file) != 0: # TQ la longueur de la file est differ
        x = file.pop(0) # on insere la valeur de l'indice 0 qu
        t.append(x[1]) # on ajoute x[1] dans le tableau t
        if x[0] != None: # si x[0] est different de None
            file.append(x[0]) # on l'ajoute a la file
        if x[2] != None: # si x[2] est different de None
            file.append(x[2]) # on ajoute x[2] dans la file
    return t # on renvoie t
```

4.4 - Utiliser un GIT ? Et tchatGPT ?

```
def parcours_largeur(AB: 'Arbre Binaire') -> list:
```

```
    """
```

Fonction pour effectuer un parcours en largeur d'un arbre binaire.

Paramètres:

AB (tuple): Un tuple représentant un nœud d'arbre binaire, où AB[0] est le nœud racine et AB[2] est le fils droit.

Retourne:

list: Une liste contenant les valeurs des nœuds visités dans l'ordre de parcours en largeur.

Exemple:

```
>>> AB = ((None, 1, (None, 2, None)), 3, (None, 4, None))
```

```
>>> parcours_largeur(AB)
```

```
[3, 1, 4, 2]
```

```
    """
```

```
    ...
```

4.4 - Utiliser un GIT ? Et tchatGPT ?

```
def parcours_largeur(AB: 'Arbre Binaire') -> list:
```

```
    file = [AB]
```

```
    t = []
```

```
    while len(file) != 0:
```

```
        x = file.pop(0)
```

```
        t.append(x[1])
```

```
        if x[0] != None:
```

```
            file.append(x[0])
```

```
        if x[2] != None:
```

```
            file.append(x[2])
```

```
    return t
```

```
# Initialise une file avec le nœud racine
```

```
# Initialise une liste vide pour stocker
```

```
# Continue le parcours tant que la file
```

```
# Récupère le premier nœud de la file
```

```
# Ajoute la valeur du nœud récupéré à la liste
```

```
# Si le fils gauche existe, l'ajoute à la file
```

```
# Si le fils droit existe, l'ajoute à la file
```

```
# Retourne le résultat du parcours
```

4.5 - Mise en place de rituels au début du cours

Voici quelques exemples :

- ▶ rituel 1
- ▶ rituel 2
- ▶ rituel 3
- ▶ rituel 4
- ▶ rituel 5

4.6 - Réflexions sur l'épreuve

Reste à :

*écarter les sujets problématiques pour diverses raisons (algorithme vu pendant la formation, ou pas ? Code ne respectant pas les bonnes pratiques, ...)
se mettre d'accord sur ce qui est attendu, accepté, ou pas. . .*

Pour ça :

CODIMD

5 - Le Grand Oral

Voyons si on peut se mettre d'accord sur :

..

un sujet faisable en 3e, c'est un sujet de NSI ?

..

un sujet qu'on ne peut rattacher à un cours de NSI, c'est ok ?

..

Bref, qu'est-ce qu'on accepte comme sujet pour nos élèves avant de les envoyer dans le grand bain.

6 - présentation de projets

Explication : revue de projets cf vademecum

6.1 - revue de projets

Une façon de faire ? Demander un compte-rendu systématique d'actions à chaque séance et notamment des fiches récapitulatives à certaines étapes cruciales :

- ▶ l'étape de réflexion préalable : découpe du problème en sous-problèmes et affectation d'une tâche à certaines fonctions dont on va définir au moins le nom.
- ▶ prototypage des fonctions
- ▶ répartition des tâches entre les différents membres de l'équipe.

Stocker l'ensemble des travaux via l'ENT ou un GIT est certainement la méthode la plus stable et celle qui permet de noter la progression du travail, et surtout : qui permet de vérifier que le travail n'a pas été intégralement fait par une IA.

Exemples de projet

Projet maitre du temps

Projet Analyseur