1 L'évaluation

La note d'ISN au baccalauréat est donnée par une commission constituée du professeur (M.JUNIER) et éventuellement d'un autre enseignant.
L'évaluation orale individuelle d'une durée de 20 minutes se déroulera en deux temps :

- *Première partie :*

  Le candidat effectue une présentation orale de son projet, d'une durée maximale de 8 minutes, pendant laquelle il n'est pas interrompu. Il s'appuie pour cela sur un dossier-projet de 5 à 10 pages, hors annexes, élaboré à l'aide de l'outil informatique.

  Ce projet est structuré de façon à mettre en évidence :

  – le but visé et les moyens choisis pour atteindre ce but ;
  – la démarche de projet qui a conduit au résultat tel que présenté ;
  – la dimension collaborative du projet liée au travail en équipe (2 à 3 élèves).

  La note de zéro est attribuée à tout candidat ne présentant pas un dossier-projet conforme à la définition ci-dessus .

  Un dossier-projet est considéré non conforme s'il n'est pas personnel ou n'est pas réalisé avec l'outil informatique, ou comporte moins de 5 pages, hors annexes.

- *Deuxième partie :*

  La commission interroge le candidat sur différents aspects de son projet et sur son lien avec les compétences fixées par le programme, puis élargit ce questionnement aux autres compétences spécifiées dans le programme.

  La commission dispose d'une grille de compétences pour l'aider dans son évaluation (voir ci-après).

1.1 Grille de compétences et capacités mises en jeu dans l'enseignement « informatique sciences du numérique » (ISN)

Compétences			I	II	Capacités et exemples d'observables
C1	Décrire et expliquer une situation, un système ou un programme I = 1 point II= 2 points	**C1.1**		x	Justifier dans une situation donnée, un codage numérique ou l'usage d'un format approprié, qu'un programme réalise l'action attendue…
		C1.2	x	x	Détailler le déroulement d'une communication numérique, le rôle des constituants d'un système numérique, le rôle des éléments constitutifs d'une page web, ce qu'effectue tout ou partie d'un programme ou de l'algorithme associé, l'enchaînement des événements qui réalisent la fonction attendue par un programme…
C2	Concevoir et réaliser une solution informatique en réponse à un problème I = 2 points II= 3 points	**C2.1**	x	x	Analyser un besoin dans un système d'information, le fonctionnement d'un algorithme…
		C2.2	x	x	Structurer une formule logique, des données, une arborescence, une page web, une approche fonctionnelle en réponse à un besoin…
		C2.3	x	x	Développer une interface logicielle ou une interface homme-machine, un algorithme, un programme, un document ou fichier numérique…
C3	Collaborer efficacement au sein d'une équipe dans le cadre d'un projet I = 1 point II= 2 points	**C3.1**	x	x	Agir au sein d'une équipe dans des rôles bien définis, en interaction avec le professeur.
		C3.2			Rechercher et partager une information, une documentation, une explication.
		C3.3		x	Maîtriser l'utilisation d'outils numériques collaboratifs du type ENT, système de gestion de contenu (CMS), groupe de travail, forums…
C4	Communiquer à l'écrit et à l'oral I = 2 points II= 3 points	**C4.1**	x		Documenter un projet numérique pour en permettre la communication en cours de réalisation et à l'achèvement, tout en précisant le déroulement et la finalité du projet.
		C4.2	x	x	Présenter le cahier des charges relatif à un projet ou un mini-projet, la répartition des tâches au sein de l'équipe, les phases successives mises en œuvre, le déroulement de l'ensemble des opérations…

		C4.3		x	Argumenter les choix relatifs à une solution (choix d'un format, d'un algorithme, d'une interface...).
C5	Faire un usage responsable des sciences du numérique en ayant conscience des problèmes sociétaux induits	**C5.1**			Avoir conscience de l'impact du numérique dans la société notamment de la persistance de l'information numérique, de la non-rivalité des biens immatériels, du caractère supranational des réseaux, de l'importance des licences et du droit.
		C5.2			Mesurer les limites et les conséquences de la persistance de l'information numérique, des lois régissant les échanges numériques, du caractère supranational des réseaux.

1.2 Fiche d'évaluation de l'épreuve ISN en cours d'année

Première partie : Évaluation d'un projet et soutenance orale (notée sur 8 points).

Compétences	Notation	Capacités mises en jeu
C1	notée sur 1 point	C1.2
C2	notée sur 2 points	C2.1, C2.2, C2.3
C3	notée sur 1 point	C3.1
C4	notée sur 2 points	C4.1, C4.2
Globalisation	notée sur 2 points	
Total	= /8	

Seconde partie : Dialogue argumenté (noté sur 12 points)

Compétences	Notation	Capacités mises en jeu
C1	notée sur 2 points	C1.1, C1.2
C2	notée sur 3 points	C2.1, C2.2, C2.3
C3	notée sur 2 points	C3.1, C3.3
C4	notée sur 2 points	C4.2, C4.3
Globalisation	notée sur 3 points	
Total	= /12	

2 Le contenu du dossier projet

Vous allez conduire votre projet par équipe de 3 ou 2 (par exemple quatre équipes de 3 et deux équipes de 2).
Vous devez réaliser une application informatique à l'aide de scripts écrits en Python. L'application doit être exécutable sur tout type de plateforme (Mac-OS, Linux, Windows). Elle peut s'accompagner d'extensions (documents de présentation ...) qui peuvent aussi comporter des pages web en HTML/CSS. Cependant les programmes en Python doivent représenter le coeur du projet.

Les fichiers sources des codes Python pourront être demandés par le professeur pendant la phase de développement. Ils devront respecter les conditions suivantes :

- être encodés en UTF-8
- être bien documentés
- présenter un découpage du code en fonctions ou modules facilitant sa lisibilité et sa maintenance

Votre dossier-projet devra aborder les thèmes suivants :

☞ Les **buts** poursuivis et les motivations (culture informatique, goûts personnels ...).

☞ Le **cahier des charges** : quelles réponses doit apporter le programme, sous quelle forme (mode graphique ou mode texte ?), selon quelle interface utilisateur ? ...

Ce cahier des charges doit d'abord être réalisé sous la forme d'un **Pad**, visible par l'enseignant et réalisé avec l'application `framapad` de `Framasoft`.

`https://framapad.org/`

☞ Le **synopsis** de l'architecture du projet sous forme de **carte mentale**, visible par l'enseigant et réalisée avec l'application `framindmap` de `Framasoft`.

`https://framindmap.org/c/maps/`

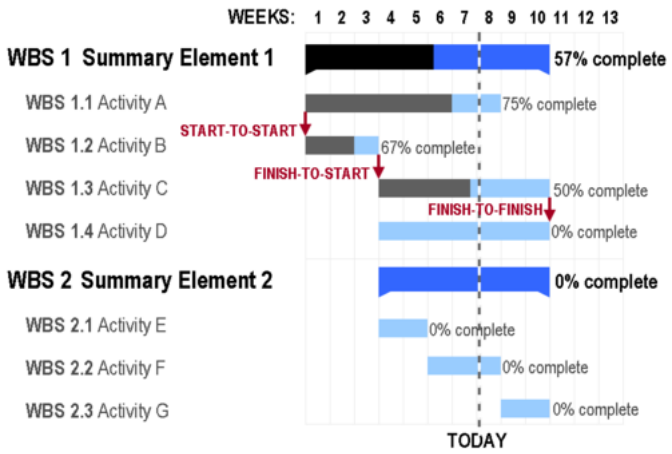
☞ Une description plus détaillée de **l'algorithmique** du programme concrétisant le projet :

➟ structures de données utilisées
➟ architecture de l'application : programme principal, découpage en fonctions ou en modules ...
➟ description d'un ou plusieurs scénarios possibles d'exécution

☞ Une présentation de **l'environnement de travail** : Integrated Development Environment (Pyzo, Spyder ...), bibliothèques Python utilisées (PIL, Tkinter, Pygame), paradigme de programmation (procédural, objet ...) ...

☞ La **dimension humaine** du projet : répartition du travail (diagramme de Gantt voir la figure ci-dessous), planning, échanges au sein du groupe, utilisation d'outils numériques de travail collaboratif (webmail, Dropbox, Framapad, Framadrop, Framindmap ...)

☞ Le **bilan** du projet avec les difficultés rencontrées, les points positifs/négatifs, les prolongements possibles du projet, l'apport humain ...



Votre dossier-projet devra respecter les conditions de mise en forme suivantes :

- Il doit comporter au minimum 5 pages hors-Annexes en police de taille 10 point avec interligne simple.
- Les codes sources, images etc . . . sont considérés comme des Annexes.
- Des jeux de tests (avec éventuellement des captures d'écran) doivent être fournis en Annexes pour chaque composant essentiel de l'application.
- Il doit être réalisé à l'aide de l'outil informatique dans des formats de fichiers ouverts (pas de document Word).
- Il doit être remis sous formats papier et numérique dans les délais (au moins une semaine avant l'évaluation) aux membres du jury.

3 Quelques règles pour réussir son projet

Source : *ISN Activités numériques de programmation en Python* de Frédéric Laroche chez Ellipse.

- **Règle n°1** **Je comprends ce que fait mon programme et je suis capable de le refaire**.

  Il faut choisir un projet à sa portée sur les plans techniques et algorithmiques.

  Lors de l'examen final la commission évaluera la compétence C1 « *Concevoir et réaliser une solution informatique en réponse à un problème* » et la compétence C2 « *Décrire et expliquer une situation, un système ou un programme* ».

- **Règle n°2** **Je choisis un projet simple mais pouvant se développer ultérieurement**.

  En l'informatique les projets marquants ne sont pas figés, leur code doit être suffisamment documenté et modulable pour permettre des développements futurs éventuellement par d'autres personnes que les initiateurs du projet.

  Pour faciliter la diffusion des savoirs, il faut développer votre projet dans l'esprit des logiciels libres même si votre programme est trop modeste pour nécessiter une licence de type GPL.

- **Règle n°3** **J'évite les problèmes techniques difficiles à maîtriser et j'essaie d'envisager la situation dans sa généralité**.

  Avant de coder, il faut passer par l'étape papier + crayon et réaliser un synopsis du projet : cas d'utilisations, actions réalisées, modèle de structures de données (types de données Python, fichiers . . .), algorithmique (boucles, fonctions). Ainsi on peut valider les différentes étapes au cours de l'avancement du projet et disposer toujours d'une vue de ce qu'il reste à réaliser.

  Notre but est l'apprentissage de la programmation, l'utilisation de bibliothèques externes (PIL, Pygame . . .) ou de techniques de programmation sophistiquée (programmation objet) doit être justifiée. Pour toute fonction de bibliothèque qu'on utilise, on doit être capable de donner les paramètres d'entrée, l'action réalisée, les paramètres de sortie et un exemple d'utilisation contextualisée.

- **Règle n°4** **Je ne perds jamais de vue mes objectifs et je ne travaille pas seul.**.

  Lorsque le synopsis du projet est prêt, on peut remplir une fiche de suivi comme celles proposée en Annexes 1 et 2 avec le cahier des charges, le planning et la répartition des tâches.

  De plus il est recommandé de tenir à jour « un cahier de suivi » (sous format papier ou numérique) en y incorporant des fiches d'état du projet comme celle proposée en Annexe 3.

  Il faut aussi mettre en place un espace de partage électronique de documents (type Dropbox . . .) pour faciliter l'échange d'informations au sein du groupe.

  Enfin il est nécessaire de faire le point régulièrement avec le professeur.

4 Quelques idées de projet

La plupart des propositions sont tirées du manuel Informatique Et Sciences du Numérique de Gilles Dowek aux éditions Ellipse.

- **Un générateur d'exercices de calcul mental**

  Programmer un générateur d'exercices de calcul mental : le programme choisit aléatoirement une opération et deux nombres, et vérifie la réponse de l'utilisateur. On peut ensuite poser une série de questions et compter le score total. On pourra enfin prévoir plusieurs niveaux de difficulté selon les opérations proposées ou la taille des nombres à calculer, et laisser l'utilisateur choisir son niveau de difficulté ou attribuer des scores variables aux réponses.

- **Un jeu éducatif pour les petits (apprentissage du calcul ou de la lecture . . .)**

  La suite éducative `Gcompris` téléchargeable pour toutes les plateformes sur ce site `http://gcompris.net/index-fr.html` propose plein d'exemples de petits jeux éducatifs dans différents domaines (mathématiques, lecture, sciences physiques . . .). Vous pouvez essayer de programmer votre propre version de l'un des jeux ou créer un fork !

- **Le jeu de la vie**

  Sur un damier carré, on dispose des créatures de manière aléatoire. La population évolue d'un état au suivant selon les règles suivantes :

  – une créature survit si elle a 2 ou 3 voisines dans les 8 cases adjacentes et elle meurt à cause de son isolement ou de la surpopulation sinon,

  – une créature naît dans une case vide s'il y a exactement 3 créatures dans les 8 cases voisines, et rien ne se passe dans cette case sinon.

  Écrire avec la bibliothèque Tkinter de Python un programme qui simule le développement d'une population.

- **Générateur d'oeuvres aléatoires**

  Trouver, sur le site web d'un musée, des tableaux. Choisir aléatoirement un petit détail d'un de ces tableaux et agrandir ce détail en un nouveau tableau. Changer les couleurs, le contraste, mélanger des détails issus de deux tableaux différents, etc . . . Vérifier la licence de ces images et comprendre s'il est possible de publier ses résultats ou non.

- **Qui est-ce ?**

  Créer une version numérique et graphique du jeu de société « Qui est-ce ? ». Quelle est l'utilité de la notion de dichotomie pour jouer à ce jeu ?

- **Jeu des 36 chandelles**

  Au départ, les deux joueurs disposent de 36 chandelles allumées. Tour à tour, ils doivent en éteindre de une à six. Celui qui éteint la dernière a perdu. Programmer ce jeu. Programmer une stratégie permettant à l'ordinateur de gagner à coup sûr (source : Jeux & Stratégie n°5).

- **Tours de Hanoi**

  On considère trois tiges nommées A, B et C de gauche à droite.
  Au début du jeu, n anneaux de diamètres distincts sont empilés sur la tige la plus à gauche A par ordre croissant des diamètres de bas en haut de la tige.
  Le but du jeu est de déplacer les n anneaux de la tige A vers le tige la plus à droite A en un nombre minimal de déplacements et en respectant les règles suivantes :

  – on ne peut déplacer qu'un disque à la fois ;

  – on ne peut pas poser un disque plus grand sur un disque plus petit.

  – on peut utiliser les trois tiges pendant le jeu.

  Ecrire un programme qui résout le problème en affichant les déplacements successifs effectués.

  Réaliser ensuite une interface graphique qui permet à l'utilisateur de jouer et qui affiche la solution optimale.

- **Jeu de Tic-Tac-Toe**

  Le Tic-Tac-Toe est un jeu où deux joueurs placent à tour de rôle l'un des ronds O et l'autre des croix X sur un plateau de trois cases sur trois cases, jusqu'à ce que l'un des joueurs ait aligné trois symboles ou que les neuf cases soient remplies. C'est le joueur O qui commence. Décrire ce jeu sous forme d'un diagramme d'états-transitions (faire une recherche documentaire sur cette notion). Combien y a-t-il d'états possibles ? Un état du jeu peut être gagnant pour O, gagnant pour X, match nul ou en cours de jeu. Exprimer chacun des états à l'aide d ?un nombre entier. Construire un tableau qui pour chacun de ces états indique s'il est gagnant pour l'un des joueurs, nul ou en cours de jeu et, dans ce cas, les transitions possibles pour chacun des joueurs. Calculer ensuite, de proche en proche, les états dans lesquels

  – le joueur O est certain de gagner,
  – le joueur X est certain de gagner,
  – le joueur O est certain de gagner ou de faire match nul,
  – le joueur X est certain de gagner ou de faire match nul.

  Écrire un programme qui joue contre un joueur humain.

- **Algorithmes de Dijskstraa et de Floyd-Marshall**

  Faire une recherche documentaire sur ces algorithmes de recherche de plus court chemin dans un graphe puis les programmer avec une structure de données appropriée

- **Génération de labyrinthe parfait**

  Définir un labyrinthe parfait et écrire un programme de génération et de résolution de labyrinthe parfait..

- **Algorithme de sortie d'un labyrinthe**

  Faire une recherche sur l'algorithme de Pledge sur le site Interstices :

  `https://interstices.info/jcms/c_46065/l-algorithme-de-pledge`

  Programmer cet algorithme, réaliser une première version avec Rurple puis une autre avec `pygame` ou `tkinter`.

- **Déchiffrer automatiquement un message codé.**

  Écrire un programme qui déchiffre automatiquement un message codé selon la méthode de César sans connaître la valeur du décalage a priori. Rechercher ce qu'est le chiffre de Vigenère, et écrire un programme qui code un message selon ce principe. Rechercher une méthode pour décoder un message codé selon ce principe sans connaître la clé utilisée, et écrire un programme qui effectue ce décodage.

- **Daltonisme.**

  Rechercher sur le Web ce qu'est le daltonisme et quelles en sont les différentes formes. Écrire un programme qui lit une image dans un fichier au format PPM et l'affiche à l'écran comme la verrait une personne atteinte de chacune des formes de daltonisme.

- **Correcteur orthographique**

  Réaliser un correcteur orthographique. Un tel programme prend en entrée un fichier texte, le découpe en mots, cherche chaque mot dans un dictionnaire et donne la liste des mots qui ne sont pas dans le dictionnaire.

- **Bataille navale**

  Écrire un programme de bataille navale avec plusieurs bateaux. On pourra utiliser la bibliothèque Tkinter pour Python.

- **Simulateur de phénomène physique**

  Rechercher dans son livre de Physique, un phénomène physique qu'il serait intéressant de modéliser informatiquement avec une visualisation (chocs élastiques de boules particules soumises à la gravité, puis rajouter des paramètres comme la pression, la température . . .)

- **Jeu de Pong**

  Écrire une version Python du jeu Pong (ou de Space Invaders, Arkanoid, PAcman, Donkey Kong ou d'un classique du jeu vidéo . . .) On pourra utiliser les bibliothèques Tkinter ou Pygame pour Python.

- **Mastermind**

  Écrire un programme qui lit deux tableaux de quatre éléments au clavier et indique le nombre d'éléments en commun dans ces deux tableaux. Écrire un programme qui joue au Mastermind : tire au hasard la la combinaison secrète et répond aux propositions de l'autre joueur.

  Prolongement possible : remplacer le joueur par l'ordinateur.

- **Jeu de pendu**

  Réaliser une version du jeu de Pendu avec interface graphique où le joueur joue contre l'ordinateur.

- **Jeu de puissance 4**

  Réaliser une version du jeu de Puissance 4 pour 2 joueurs avec interface graphique.

  Prolongement possible : un joueur joue contre l'ordinateur.

- **Codes barres**

  Rechercher des informations sur les codes barres EAN13 utilisés sur les produits de consommation courante. Construire un programme générant une image du code barre au format PBM à partir du code numérique.

- **Traitement d'images**

  Ecrire une bibliothèque de traitement d'images (filtres de convolution, filtre médian pour réduction du bruit, pixellisation pour floutage, filtres de détection de contour par érosion/dilatation) et donner quelques exemples d'application (par exemple comment déterminer l'âge d'un arbre à partir d'une image d'une section de son tronc). Dans le même domaine, on peut explorer les thèmes de la stéganographie, du tatouage d'images . . .

- **Cryptographie et envoi de mail**

  Écrire une bibliothèque implémentant des algorithmes déchiffrement/déchiffrement à clefs (César, Vigenère).

  Écrire ensuite une petite application Python, émulant le fonctionnement du logiciel Pretty Good Privacy apparu dans les années 1990, qui a permis l'authentification de l'émetteur par un système de clefs publiques/clefs privés (cryptographie asymétrique basée sur l'algorithme RSA) et l'échange sécurisé de clefs pour le chiffrement des messages entre deux destinataires (pour simplifier on utilisera les fonctions basique de chiffrement comme César ou Vigenère).

- **Cryptographie, méthode de Fleissner**

  Écrire une petite application graphique simulant la méthode de cryptographie de Fleissner.

  `http://www.bibmath.net/crypto/index.php?action=affiche&quoi=ancienne/fleissner`

- **Un agenda électronique**

  Réaliser une application qui peut afficher un calendrier mensuel et qui permet d'associer à un jour une ou plusieurs tâches, de lister les tâches associées à un jour . . . On pourra utiliser la bibliothèque Tkinter pour l'interface graphique et des fichiers textes pour le stockage des tâches.

  On peut aussi envisager la réalisation d'une application web en utilisant l'interface CGI du serveur Apache installé sur la clef ISN et une base de données SQL ou en utilisant le framework Django.

- **Une application pour le journal du lycée (ou pour le ciné club)**

  Réaliser une application qui permet de gérer les articles du journal du lycée :

  – enregistrement d'articles selon auteur, catégorie, date de publication
  – recherche par critères dans la base de données ou les fichiers textes où sont stockées les données ...

- **Une solution d'affichage déportée pour les écrans d'information du lycée**

  Offrir la possibilité aux secrétaires de paramétrer un affichage sur les écrans d'information depuis leur bureau : accès à distance par le réseau et gestion de l'affichage à l'aide d'un Raspberry Pi (possibilité d'achat du matériel par le lycée).

- **Réalisation d'une calculatrice**

  Ecrire un programme d'émulation de calculatrice qui peut fonctionner selon deux modes :

  – en mode infixe, elle peut évaluer les expressions correctement parenthésées, utilisant les quatre opérations de base, en notation infixée (fonction `evl` interdite).
  – en mode postfixe, elle peut évaluer les expressions en notation polonaise inverse.

  Une fonction de conversion doit permettre de traduire en notation infixée une expression postfixée et vice-versa.

  Une interface graphique doit permettre également le tracé de graphiques avec réglage de la fenêtre.

- **Sudoku**

  Réaliser un solveur de Sudoku avec ou sans interface graphique.

5 Quelques bibliothèques de Python qui pourraient vous être utiles

☞ Réalisation d'interfaces graphiques plutôt statiques : **tkinter** (sous Python 3) ou **Tkinter** (Python2) : `http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/index.html`

☞ Traitement d'images : **PIL** ou son fork **Pillow** : `https://pypi.python.org/pypi/Pillow/`

☞ Réalisation d'interfaces graphiques plutôt dynamiques (jeux) : **pygame** : `http://pygame.org/news.html`

☞ Bibliothèques pour le calcul scientifique : **numpy** ou plus généralement **scipy** : `http://scipy.org/`

☞ Réalisation de graphiques pour les mathématiques ou la physique (possibilité d'animation) : **matplotlib** : `http://matplotlib.org/`

☞ Réalisation de pages web dynamiques :

➟ **Django** (Python2 uniquement) : `http://www.django-fr.org/`
➟ **CherryPython** : `http://www.cherrypy.org/`

6 Quelques applications pour la gestion de projet ou le travail collaboratif

☞ Rédaction de documents à plusieurs (avec possibilité de chat) : **Framapad** accessible depuis `https://framapad.org/`.

☞ Rédaction de feuilles de calculs à plusieurs : **Framacalc** accessible depuis `https://framacalc.org/`.

☞ Réalisation de carte mentale : **Framindmap** accessible depuis `https://framindmap.org/c/maps/`.

☞ Echange de fichiers volumineux de façon anonyme et sécurisée (temps limité) : **Framadrop** accessible depuis `https://framadrop.org/`.

☞ Partage d'images de façon anonyme et sécurisée (temps limité) : **Framapic** accessible depuis `https://framapic.org/`.

☞ Partage de fichiers (sans limitation dans le temps) : **Dropbox** accessible depuis `https://www.dropbox.com/`.

ANNEXE 1 : Fiche de suivi du projet / Cahier des charges

Date	Description	Etat actuel rien/en cours/fini
Descriptif détaillé		
Référence à une réalité/Contexte		
Documentation et commentaires du programme		
Références / Sitographie		
Algorithme principal		
Découpage du code en fonctions ou modules		
Utilisation d'une bibliothèque		
Structure de données		
Gestion de fichiers		
Interface utilisateur		

ANNEXE 2 : Fiche de suivi du projet / Planning

Date	Etape du projet (Que faire ?)	Outils (Comment ?)	Qui fait quoi ?

ANNEXE 3 : Fiche d'état du projet

Groupe	
Date :	**Fiche n°**
Etape du projet (selon le planning établi)	
Bilan : (où en est le projet ?)	
Eventuelle(s) difficulté(s) : (le projet est-il freiné ou bloqué, pourquoi ?)	
Solutions proposées : (Comment surmonter cette difficulté ? Qui fait quoi ?)	
Prochaine étape du projet : (ce que chacun doit faire fait et pour quand)	
Observations particulières	