1.Configurateur de Planète (Planetmaker)
Utilisation : très simple, intuitif ; s'ouvre sans bibliothèque spéciale (turtle et tkinter uniquement) ce qui est la demande initiale. L'idée est originale, intéressante, ce n'est pas un jeu ; certaines idées n'ont pas abouti mais la réalisation est assez élaborée (dégradés, ellipses, formes aléatoires). On se dit à la fin « c'est tout ? ». On aurait pu imaginer un scénario plus complet (enregistrer une image, constituer une collection...) et une zone de saisie mieux structurée (elle empiète sur l'espace) mais le temps était limité et le travail s'est concentré sur les détails graphiques, ce qui n'est pas rien.
Programme : 474 lignes ; bien commenté (sans excès), variables aux noms explicites, structure fonctionnelle claire et bien conçue. Le dessin de la planète est parfois refait sur un changement, mais pas toujours (la déselection de l'anneau n'entraine pas le redessin ce qui n'est pas logique, de même, changer la dimension ou la couleur n'entraîne pas de changement... ces réglages devraient être mieux corrélés au dessin qui devrait être refait à chaque changement)
Compte-rendu : préprojet très bref ; doc final bien complet, très bien écrit et illustré, met l'accent sur les différentes idées, leurs développements et leur abandon pour certains.
Note : 17/20

2.Morpion
Utilisation : très simple, fonctionne avec la souris ; s'ouvre sans bibliothèque spéciale (turtle uniquement) ce qui est minimaliste. Les questions initiales sont un peu agaçantes (il faut lire le mode d'emploi, taper le mode de jeu (« 1 »), son niveau (« difficile ») et enfin son nom (« PM ») ce qui aurait pu se réaliser en une seule fois dans une fenêtre d'accueil. Mais c'est le seul défaut, le jeu fonctionne très bien et implémente même une intelligence artificielle sommaire très efficace.
Programme : 725 lignes ; bien commenté (sans excès), variables aux noms explicites (7 tortues!), structure répétitive : certains blocs auraient pu être regroupés dans des fonctions pour éviter les redondances, certains objets auraient pu être définis dans des listes (feurouge, feuvert et feujaune dans une liste feu à 3 éléments, de même pour les joueurs et les scores), certains tests auraient pu être synthétiques (près de 50 tests pour se décider sur le choix d'une case parmi 9 ! cela aurait pu s'écrire en 9 tests). Ce parti pris rend pénible la lecture et délicat la correction (n'a t-on rien oublié) mais on comprend bien la logique de cette écriture.
Compte-rendu : préprojet très précis, donne l'idée du rendu final ; doc final très complet, bien écrit et décrivant les différentes améliorations apportées depuis l'idée initiale
Note : 19/20

3.Street Fighter (en moins bien)
Utilisation : fonctionne avec la souris et le clavier et ne nécessite que tkinter ce qui est bien. Cependant, je vais le dire tout de suite, je ne connaissais pas ce jeu et sa découverte ici, dans cette version « en moins bien » (comme annoncé dans le titre), ne me donne pas envie d'en savoir davantage : il s'agit de combats entre 2 personnages qui s'affrontent en se donnant des coups de pied... Il y en a un qui gagne assez rapidement on peut donc recommencer mais comme il n'y a pas de score, c'est quoi le but ? Malgré mon manque d'intérêt pour ce jeu qui n'est pas très fluide ni même esthétique (les personnages donnent des coups de pied mécaniques qui touchent leur cible même sans contact...), je constate qu'il y a eu beaucoup de travail : des costumes multiples pour chaque personnage, des boutons, des quantités de texte et des images, une douzaine de touches sensibles pour déplacer les joueurs. Cela fonctionne mais le résultat est un peu décevant. J'ai apprécié les barres de vie qui sont bien construites (changent de couleur en dessous d'un certain pourcentage : vert, jaune puis rouge), les changement de costumes (même si peu réalistes) et l'effort de construction d'une documentation cohérente et riche.
Programme : 1220 lignes ! Dont près de 500 lignes pour les attaques spéciales que je n'ai pas du tout utilisé ni même perçu lors de mes essais du jeu. Les commentaires n'aident pas beaucoup puisqu'ils se limitent à décrire succintement le rôle d'une fonction, par exemple « Attaque spéciale du personnage 1 » est la seule explication pour les 250 lignes qui suivent... Du coup je ne connaitrais pas les attaques spéciales. Le programme a été développé sans interaction avec le

professeur, du coup je n'ai pas pu avoir d'aperçu ni même d'explication pendant le temps de sa réalisation. Je vois bien à peu près ce qui se fait à chaque ligne mais c'est trop me demander d'en déduire le fonctionnement de ces attaques secrètes. Pour moi, elles resteront secrètes ou plutôt cachées. Je suis stupéfait de voir qu'il y a 4 costumes dessinés par personnage et 12 personnages : ils ont vraiment été dessiné à la main ? Quel énorme travail ! En plus il est précisé que ces personnages contrastent avec ceux du jeu initial : il s'agit donc d'invention... mais quel temps y a tu consacré ? Cela ne peut pas entrer dans mon évaluation car pour moi le travail graphique n'est pas de la programmation et il s'agit d'un projet de programmation python.
Compte-rendu : préprojet honnête et réaliste ; doc final assez bien construit, mentionnant les longues tâches graphiques et textuelles (création de 12 sprites accompagnés de documentation) qui, ceci dit en passant, ne sont pas forcément utiles (2 sprites auraient suffi pour mettre au point le fonctionnement) sauf pour enrichir le projet. Des détails sur les attaques (PYRAX et MUNJA, Omega Z) révèlent d'autres subtilités pas forcément perçues par un utilisateur non-averti et pas forcément comprises en regardant le programme non commenté.
Note : 16/20

4.Sudoku
Utilisation : pas pu tester encore
Programme : le fichier rendu comme programme (extension .py.lnk) n'est pas le programme mais un export inutilisable.
Compte-rendu : préprojet ; doc final bien rédigé, élégant et assez circonstancié
Note : 9/20
Le programme m'est enfin parvenu, in extremis (4 jours avant le conseil de classe!). Il est très long (800 lignes) et bien commenté. Le fonctionnement est très bien : plusieurs mode de jeu ; l'ordi peut proposer une grille qui est créée de toute pièce pour les dimensions 4, 9 ou 16 et 3 niveaux de difficultés différents (ce facteur influe sur le nombre de chiffres enlevés) ! La conception est d'un niveau assez élevé en Python, j'imagine que l'aide de la soeur de Ilyana a été massive (vous le dites d'ailleurs) mais le résultat est impressionnant. Techniquement, une grille de Sudoku doit n'avoir qu'une seule solution possible, ce que vous ne testez pas. Votre programme de résolution (la fonction récursive « solve_tableau ») contient un mode « demo » dont vous ne parlez pas et qui se lance si, au lieu d'écrire « s = Sudoku(n, 0, False) » on écrit « s = Sudoku(n, 0, True) ». Cette fonctionnalité (pas si simple) passée sous silence est curieuse, et l'utilisation d'une fonction récursive également. Je finis par me demander ce qui constitue votre travail (au moins les commentaires vu le nombre de fautes d'orthographes, la partie turtle que j'ai vu se réaliser en classe) et ce qui a été emprunté à un programme existant (de la soeur en question ou d'une source autre non citée). Compte tenu de toutes ces remarques (retard+incertitude/doutes sur la technicité acquise par les membres du groupe) je mets une note correcte, 15/20, mais encore une fois, il s 'agit d'une côté mal taillée car si vous étiez les seules réalisatrices cela vaudrait quasiment 20/20...

5.ImageStylizer
Utilisation : très simple, si on dispose de la bibliothèque PIL (sinon l'importer) et aussi tkinter. L'interface est très efficace : on ouvre directement une image quelconque et on travaille dessus à l'aide des quelques boutons qui permettent de choisir interactivement différents mode de transformation ; l'export marche aussi très bien. Les images créées sont intéressantes. Le programme peut s'avérer utile pour qui aime modifier ses images ou peut servir de base pour l'enrichir de nouvelles fonctionnalités (déformations, pixelisation, etc.)
Programme : deux programmes sont utilisés : le principal ne fait que 72 lignes, les fonctions sont déplacées dans un 2ème programme de 230 lignes. Les fonctions sont bien commentées avec un *docstrings* (entre """…""") et des commentaires sur une ligne ou en fin de ligne (commencent par #) pour expliciter le contenu d'une variable ou l'objectif ponctuel de la partie visée. La note reflète également le comportement de support technique aux autres groupes, largement exploité par certains.
Compte-rendu : préprojet succinct mais conforme ; doc final très bien rédigé et complet avec un

mode d'emploi détaillé (ce qui n'est pas inutile) et des images donnant des exemples (ce qui est toujours agrable).
Note : 20/20

6.The Elder’s Heart (jeu de plateforme)
Utilisation : très simple, une fois installée la bibliothèque pygame. Le scénario du jeu est magnifique avec un graphisme évolué et riche (et encore je n'ai réussi pour l'instant que les 2 premiers niveaux, et il y en a 6). Les points d'avancement (*Checkpoint*, losanges avec centre changeant de couleur) sont bien utiles pour éviter de tout recommencer quand une erreur est commise. L'ouverture et la fermeture de l'espace de jeu est bien réalisée, ni trop rapide ni trop lent, le jeu est très fluide avec seulement 3 touches à utiliser. On s'habitue à ce que la marche suive les lignes et angles de rectangles alors que le visuel est à l'intérieur de ces rectangles (arbres en boules).
Programme : L'ensemble est impressionnant avec 8 fichiers de code python totalisant quelques 1800 lignes, de nombreux fichiers csv pour les 6 niveaux écrits, et de très nombreuses images pour les éléments graphiques. Les commentaires sont nombreux et explicites avec des *docstrings* (entre """…""") et des commentaires sur une ligne ou en fin de ligne (commencent par #). De très nombreuses classes (18 classes dans le seul fichier « tuiles ») permettent de structurer proprement les différents éléments.
Compte-rendu : préprojet particulièrement bien écrit et détaillé montrant un objectif très ambitieux et très bien analysé ; doc final du même tonneau, très détaillé et lucide sur les difficultés, résolues ou non. Le niveau des explications est assez technique avec des expressions du monde des jeux vidéos que je connais pas (blit l'image, fichier Tiled, pertes de frames, le dash et le saut mural...). Ce projet est vraiment excellent, largement au-dessus des attentes ce qui n'est pas un reproche.
Note : 20/20

7.NSI pixelisateur
Utilisation : utilise la bibliothèque PIL et aussi tkinter. J'essaie le programme sur mon ordi avec windows, sur Idle et Pyzo mais ça ne fonctionne pas bien : l'image ouverte n'est pas reconnue quand je la prend dans un dossier images de mon ordi (No such file or directory), il faut que je la recopie dans le dossier du programme pour qu'elle soit ouverte. Le programme fonctionne alors partiellement et produit une image au format png qui est ouverte par paint... et c'est tout... Ah non, si on prend soin de fermer la 1er image (pixelisée), il finit pas coloriser les pixels avec les choix prédéfinis. Le résultat est très bien, dommage qu'on ne peut pas travailler les couleurs ou la taille des pixels sans redémarrer le programme. Par contre les options de sauvegarde n'ont pas marché, mais comme les images s'ouvrent dans paint on peut les enregistrer facilement.
Programme : un seul programme de 300 lignes environ (beaucoup de lignes vides), relativement simple à comprendre. Quelques commentaires indiquent les fonctions des différentes parties. Ainsi j'apprends qu'il y a une fonction qui redimentionne les éléments si on réduit la fenêtre : c'est un luxe ! Le choix des couleurs me laisse perplexe : comment en choisir d'autres ? Ou plutôt pourquoi ? On ne sait pas quelle couleur sera remplacée par l'une des 18 couleurs proposées. Ce serait intéressant de montrer la palette des couleurs qui sera remplacée par chacune et ça justifierait des changements éventuels.
Compte-rendu : préprojet intéressant mais déjà très technique qui pose la question de la distance euclidienne d'une couleur à une autre alors que le projet n'est pas complètement ficelé ; doc final qui décrit les difficultés rencontrées. On regrette l'absence d'images pour illustrer le fonctionnement et il n'y a pas non plus de retour sur d'éventuelles améliorations qui pourraient être apportées à cette application.
Note : 17/20

8.EnigmeEspion
Utilisation : Un petit fichier « lisez-moi » permet de savoir ce qu'il y a à faire, c'est appréciable ; les programmes utilisent plusieurs bibliothèques (time, pygame, pytmx et pyscroll). Mis à part la

fonction time.clock()_gettime(time.CLOCK_REALTIME) de time que j'ai dû reécrire time.clock() tout simplement (sans doute un problème de mise à jour sur une bibliothèque), le jeu fonctionne très bien et même si on n'est pas trop familier avec ce type de jeu, j'ai réussi à tout faire avec un peu de temps quand même (énigmes pas trop difficiles car j'aurai pu me décourager, je me suis fait avoir avec le nombre de chevaux et le1er numéro caché dans la plage). Certains sons sont originaux (merci Léon pour ce beau morceau de trombone !) et d'autres non (le pinpon des pompiers est terrifiant, mes voisins s'en souviennent...). J'ai obtenu l'image à renvoyer à l'adresse de Léon mais je regrette presque qu'il n'y en ai pas une par niveau, par exemple un quart de l'image, afin d'envoyer à la fin les 4 images prouvant qu'on a bien tout réussi.
Programme : près de 700 lignes de code assez technique dû aux bibliothèques utilisées. Tout est bien commenté sur une ligne ou en fin de ligne (commencent par #). C'est propre ; la répartition en 6 fichiers est logique ; les images (50 images de personnages !), les éléments graphiques et les sons sont dans des dossiers spécifiques (sauf quelques images qui trainent dans le dossier principal, pas utilisées je crois). La personne qui lance le programme peut être tentée de tricher puisque tous les codes y figurent : il faudrait les dissimuler un peu mieux, par exemple en les encryptant (un fonction de décryptage peut facilement passer inaperçue)
Compte-rendu : préprojet bien détaillé et qui donne une bonne idée du travail à effectuer et des outils à employermême s'il subsiste un flou nécessaire ; le doc final apporte des compléments sur la réalisation des images de fond et des personnages, ce qui a visiblement pris beaucoup de temps vu la complexité du jeu, moins de détails est donné sur les programmes proprement dit, ce qui est tout de même l'objet du projet.
Note : 19/20

9.Tétris 2
Utilisation : Le jeu fonction à peu près, il utilise pygame ; j'ai modifié la vitesse d'avancement pour qu'il soit plus facile (niveau = 1 au lieu de 2) mais la modification de la dimension de la grille (plus large (25 au lieu de 10) et moins haute (17 au lieu de 20) rend la tâche très difficile ainsi que ça a été mentionné dans le compte-rendu. Les erreurs du jeu initial (le tetris.py copié) n'ont pas toutes été corrigées car la pièce longue continue à être présentée même si on ne la voit pas, ce qui crée des vides dans le dynamisme.
Programme : 250 lignes de code intégralement copiées de la 2ème adresse donnée (tetris.py dans https://data-flair.training/blogs/python-tetris-game-pygame/) et légèrement arrangées (traduction de certains noms anglais et ajout de 7 formes constituées de 5 carrés, ce qui rend plus difficile le jeu). Les commentaires sont absents alors qu'il y en a dans la version copiée (pourquoi les enlever?) et il est dit dans le compte-rendu que le jeu avait été bien compris... cela ne se voit pas trop du coup. La pièce à venir (à droite de la fenêtre) n'est pas changée alors que dans le programme original est l'est... Certains changements ont affecté le bon fonctionnement initial mais mon propos ici n'est pas de corriger le programme, cela aurait pu être fait en classe si une recherche sérieuse avait été faite ce qui n'a vraisemblablement pas été le cas.
Compte-rendu : préprojet ridiculement bref (4 lignes) ; doc final à peine plus long (10 lignes) mais qui donne honnêtement la source largement utilisée.
Note : 10/20
Je reviens sur ma première évaluation et augmente la note de 2 points pour les modifications apportées au programme initial qui ne sont pas si négligeables.

10.Jeu d'échecs
Utilisation : Très simple, nécessite seulement tkinter. Le jeu est aisé, fluide. Les pièces bien dessinées, celles qui sont prises apparaissent au bon endroit. Rien à dire, sauf pour les indications textuelles : « le trait est au noir » est peu visible (écrit petit) j'aurai mieux compris un zone colorée (rectangulaire ou ronde) alternativement du côté des blancs et des noirs. Bizarement lorsque le roi est en échec, ce n'est pas écrit : les pièces ne peuvent bouger sans mettre fin à l'échec mais ce serait bien, ne serait-ce que dans le message laconique « le trait est au ... » faire figurer cet échec, par

exemple avec « le trait est au … ; attention, roi en échec ! ». La promotion du pion est agréablement implémentée ; l'échec et mat bien signalé, le roque fonctionne bien. Bon, je n'ai sans doute pas tout testé ; il reste la prise en passant à réaliser mais c'est un détail, les ambitions étant largement dépassées. Je regrette à la fin d'une partie qu'il ne soit pas proposé d'en faire une autre, cela pourrait être un des développements futurs (à ajouter à la liste que vous faites dans le compte-rendu) ; on peut aussi imaginer l'affichage d'un chronomètre pour le temps de reflexion de chaque joueur...
Programme : moins de 600 lignes de code pour régler tous les mouvements possibles et ils sont nombreux, les prises, le roque, la promotion du pion, l'échec, le pat, le mat... la seule classe « Piece » ne compte pas moins de 26 fonctions ! Le code est propre, très richement commenté et les variables sont nommées de façon à refléter leur usage.
Compte-rendu : préprojet bien défini aux objectifs raisonnables, avec une liste des premières questions à résoudre ; doc final bien rédigé montrant des images de différentes phases de jeu et listant difficultés rencontrées, solutions trouvées et développements possibles.
Note : 20/20

11.Styx
Utilisation : Pas évident quand on n'a pas de notice (touche « z » pour sauter), nécessite seulement tkinter. La page d'accueil est très bien ; le bouton « quitter » semble un peu inutile quand on commence... Les sauts de la grenouille ne sont pas visibles, rien n'indique qu'on a bien changé de nénuphar (sauf l'image des nénuphars qui change), il n'y a même pas de score affiché (il est affiché quand on a perdu) alors que c'est simple à faire. La perspective n'aide pas à imaginer qu'on saute, il faudrait faire plus petits les nénuphars qui sont plus loin, éventuellement imaginer un déplacement de la grenouille sur le 2ème nénuphar et seulement progressivement décaler vers le haut la caméra.
Programme : 172 lignes de code. Il y a des commentaires, peu mais il y en a. Les initialisations sont faites un peu partout au début, au milieu, à la fin... Il y a 2 classes et des fonctions, tout cela entremélé. Il faudrait mettre de l'ordre là-dedans. Je ne suis pas sûr que le binome a bien fonctionné, le fait d'être dans 2 groupes différents exigeait une très bonne communication ce qui n'était pas toujours réalisé (l'avancement de l'une n'était pas transmis automatiquement à l'autre). L'élaboration d'un projet nécessite sans arrêt une reflexion commune sur ce qu'il y a à développer ou corriger.
Compte-rendu : préprojet honnête montrant l'idée à réaliser ; le doc final n'est pas très clair notamment la phrase « l'importation d'images se complique si d'autres fenêtres contenant l'image ne sont pas fermées n'étant même pas ouvertes réellement. Ce petit soucis se résous en ouvrant une fenêtre sans images qui ouvrira celles qui ne s'etaient pas ouvertes réellement. En les fermant, on peu relancer notre programme sans soucis ». Il y est mentionné un certain nombre de regrets par rapport à l'idée initiale (bugs non résolus, fluidité peu satisfaisante, niveaux et tutoriel non réalisés). Je ne vois pas pourquoi ce tutoriel regretté n'a pas été écrit, ne serait-ce que pour indiquer ce qu'on est censé faire...
Note : 15/20

12.Jeu de rapidité
Utilisation : Très simple, nécessite seulement tkinter. La fenêtre est très bien réalisée avec son menu permanent à gauche et les images qui s'affichent en étant bien proportionnées à droite. Il suffit de cliquer successivement sur les paires de deux images semblables, ce qui est du niveau d'un enfant de 5 ans mais pourquoi pas. Quand on a fini dans les temps, on a juste le message « le temps s'est écoulé », ce qui est assez frustrant. On s'attend plutôt à un « Bravo ! » et un « Perdu... » dans le cas contraire. On devrait pouvoir recommencer le même niveau mais l'ancien tableau d'images n'est pas effacé. C'est un bug qu'il aurait fallu identifier et corriger.
Programme : plus de 500 lignes de code. Les 38 images sont en vrac dans le dossier (ce serait mieux de les regrouper dans un dossier images). Les instructions sont très répétitives, il faut utiliser des fonctions plus généralistes, plutôt que d'adapter 3 fois une fonction spécialisée. De même il faut utiliser des listes pour les images, pour les boutons plutôt que de donner des noms spécifiques. Pour le redimensionnement des images, vous utilisez une grande et une petite alors qu'on peut facilement redimensionner avec tkinter (voir du côté des méthodes subsample() ou zoom() ou bien utiliser également la bibliothèque PIL).

Compte-rendu : préprojet assez clair qui montre une indécision (d'abord l'idée de pixeliser une image puis de faire un jeu Memory, au final ce sera un autre jeu que Memory) ; je n'ai pas vu de doc final ! C'est bien regrettable d'autant qu'il manque l'explication de l'abandon du projet initial qui visait à faire une jeu de Memory, ce qui est tout de même plus challengeant.
Note : 15/20
Après avoir reçu enfin votre compte-rendu, je constate que vous avez identifié correctement les erreurs/difficultés ; la maîtrise des outils n'a pas été suffisante pour réaliser le projet initial (retourner une carte revient à changer d'image et effacer un jeu pour le recommencer est parfaitement possible). J'augmente de 1 point la note (pour le CR).

13.Python Racing©
Utilisation : Très simple, nécessite seulement tkinter. La page d'accueil est correcte, même si pas très esthétique (toute la place est laissée à la photo, les boutons ont tous des tailles différentes, seuls les emplacement sont bien choisis) ; elle permet de changer le nom des joueurs, personnaliser à minima son véhicule (la couleur uniquement, avec un choix de 3 couleurs), modifie la vitesse des voitures et indique le mode d'emploi. Petit gadget : la souris change de forme sur 3 boutons. Le jeu n'est pas difficile, c'est juste une question de rapidité pour cliquer sur son bouton de souris (le fait que les deux joueurs partagent la même souris, l'un au clic droit l'autre au gauche n'est pas particulièrement gênant). La piste étant très courte et les voitures avançant vite, il s'agit d'un démarrage de course (mode *dragster*). Le design des véhicule est vraiment très minimaliste (2 carrés pour les roues...), rien de la voiture de course (on peut faire avancer une image avec tkinter, c'est un basique), et vu la photo de l'écran d'accueil on s'attend à mieux ! Le nom du joueur suit la voiture, ce qui n'est pas forcément utile ni réaliste (le nom peut être rappelé dans un encadré). On regrette presque d'ailleurs qu'il ne soit pas possible de jouer à plus de deux ; il aurait fallu une touche par joueur. De même, le chronomètre serait bienvenu (on peut mesurer le temps en nonsecondes, je ne sais pas où vous avez lu qu'il n'y a que des secondes entières). On a un score entre les joueurs et c'est tout ; d'ailleurs si on veut changer la vitesse sans remettre à jour le compteur, c'est possible : du coup le message à la fin de la partie est incorrect, on devrait dire quelque chose comme « revenir à l'écran d'accueil (a) ou recommencer (r)? ».
Programme : 300 lignes de code mais 100 au moins sont vides, certaines répétitions pourraient être confiées à des fonctions pour alléger encore l'écriture. Sinon, il y a des commentaires des deux types (*docstrings* assez limités et commentaires sur une ligne). Les fonctions et les déclarations de variables sont disposées alternativement, ce qui n'est pas une bonne pratique. Il y a des fonctions dans des fonctions et cela jusqu'à 3 niveaux d'imbrication (compteur() dans logijeu() qui est dans menu()...). Je ne vois pas l'intérêt de faire cela. La maîtrise de tkinter est partielle ce qui a limité les fonctionnalités et l'esthétisme du jeu.
Compte-rendu : préprojet succinct qui cerne l'idée ; doc final avec beaucoup de copies d'écran, un texte un peu approximatif et assez bref qui retrace l'évolution du projet.
Note : 16/20

14.Le pays du chiffrement
Utilisation : Le programme n'est pas bien adapté à mon ordinateur qui fonctionne sous Windows, ce programme ne nécessite que tkinter, il présente un écran d'accueil contenant un menu. Cet accueil ne contient pas toute l'image ni tout le texte. Ce problème de dimensionnement se retrouve avec le label sur le code « avocat » (quel intérêt?) qui est coupé. Le bip du code morse ouvre et ferme des fenêtres de terminal sans émettre de son. Peut-être faut-il changer de système d'exploitation ? Sinon le codage et le décodage fonctionne bien, pour ce que j'ai pu voir. Le décodage du code César propose les déchiffrages les plus probables et sinon le bouton « Montrer » (pourquoi montrer et non autres propositions?) en propose d'autres, mais curieusement ça ne marche toujours : la phrase soulignée codée avec un décalage de 12 est « zs dfcufoaas b'sgh dog pwsb orodhs o acb cfrwbohsif eiw tcbqhwcbbs gcig kwbrckg » qui n'est pas décodé malgré les 14 propositions données... Petit problème dans les statistiques ? J'ai bien aimé le code lapin que je ne connaissais pas, le braille (très

utile dans de rares cas), le code des rois de France, non c'est très bien. La partie informative « un peu d'histoire » n'est pas complète à mon goût même si ce n'est pas l'objet principal du projet.
Programme : 1300 lignes de code, et pas une seule ligne vide ! Ah si, pardon, il y en a 10. Beaucoup de lignes sont écrites pour ajouter ou supprimer les éléments des différentes options du menu ce qui n'est absolument pas intéressant (bien que nécessaire si on ne connait pas d'autre solution pour changer le contenu de la fenêtre). Pourquoi ne pas extérioriser tout ça dans un second fichier qu'il suffirait d'appeler avec des fonctions comme « installe_code_cesar() » (en important le fichier) qui ferait gagner 60 lignes, à multiplier par les 12 blocs de ce type (soit 720 lignes de moins!)
Compte-rendu : préprojet assez précis, expliquant le code César et le principe du décodage avec statistiques, ce qui a largement été dépassé par le projet final ; doc final expliquant le principe de certains codes, ce qui aurait pu figurer dans la fenêtre « un peu d'histoire » qui aurait ainsi donné une information plus complète.
Note : 18/20

15.Traitement d'image
Utilisation : Ce programme travaille avec des images au format pgm (!), un format pour le noir et blanc dans lequel ne sont jamais les images courantes. J'ai essayé de convertir des photos normales (jpeg) dans ce format avec Gimp et IMDisplay, 2 logiciels qui le permettent mais les images converties ne sont pas acceptées par le programme. J'ai ensuite essayé le site proposé (filext.com) mais ça n'a pas marché non plus (message d'erreur : 'charmap' codec can't decode byte 0x81 in position 432: character maps to <undefined>). Du coup, il me reste les 2 petites images proposées (il y en a tellement de plus belles sur internet) pour tester le programme. L'interaction se fait sur la console (le programme ne nécessite que la bibliothèque os), le fichier enregistré au format pgm est bien une image qui a bien subi la transformation demandée mais les 2 échantillons sont tellement petits qu'on ne voit pas trop l'intérêt du programme. L'interaction avec la console est limitée puisqu'une fois l'image modifiée, le programme s'arrête. On s'attendrai à une possibilité de recommencer, à une visualisation des 2 images, mais non, il faut ouvrir l'image modifiée (avec un logiciel capable de lire de telles images) pour la voir.
Programme : 250 lignes environ assez bien commentées bien que ce soit irrégulier et dégressif (1ère fonction dotée d'une *docstring* de 10 lignes, les 3 suivantes d'une ligne seulement, puis les autres 0 ligne). Les traitements d'image sont intéressants (floutage, photomaton, rotation, inversion,...) bien que le format employé (pgm) et l'accès aux teintes soient inhabituels (juste une variable par pixel). Une instruction passée en commentaire me met « la puce à l'oreille » (#if __name__ == "__main__":), je me demande si ce n'est pas simplement le plagiat d'un programme existant sur le net. Je passe le programme à un site anti-plagiat. J'ai aussi des doutes car je ne me souviens pas avoir vu le groupe travailler avec acharnement sur le projet ni poser de questions. Le sujet est par ailleurs étudié en seconde en SNT et on trouve pas mal d'exemples de programmes similaires, pas toujours en python, pas toujours limité au format PGM et rarement sans utiliser PIL. Je n'ai pas retrouvé non plus les vilaines petites photos proposées mais je suppose que ce ne sont pas des photos de vacances... je dois en conclure même si ça me paraît incertain qu'il y a eu un certain travail, difficile à évaluer.
Compte-rendu : préprojet très succinct indiquant que le projet fera diverses actions sur une image ; le doc final est à peu près aussi bref et peu précis. Exemple « Il faut convertir l'image en un autre format pour l'ouvrir » : Quelle image ? Quel format ? Qui ouvre l'image (le programme ou l'utilisateur)? Pourquoi n'y a t-il pas d'exemples dans le compte-rendu ?
Note : 12/20

16.Reproduction du T-Rex Game
Utilisation : Super jeu qui plaira à mon fils (ressemble un peu à geometrydash qu'il affectionne), moi je coince assez rapidement et pour recommencer 1000 fois il faut avoir le temps. Il y a de la musique, un rythme soutenu, des graphismes très bien réalisés. On dirait presque un travail de pro or je n'ai pas souvent eu l'occasion de voir le groupe travailler à ce projet.

Programme : 260 lignes de code ; les bibliothèques importées sont pygame et os. Cinq ou six classes sont définies, le travail de programmation est réalisé de main de maître, avec une maîtrise totale de pygame, pour des élèves qui situaient au début de l'année leur niveau en programmation à 10 sur 20... c'est louche. Comme je suis un peu irrité par le projet précédent, je reste sur ma veine parano et soumets le fichier à mon détecteur de plagiat. Le résultat ne se fait pas attendre : de l'UNIVERSIDAD NACIONAL DE UCAYALI (au Perou), un jeu (CARSCIVIL) est donné où le dinosaure est un « duck », le petit cactus et le grand cactus un « SmallBlock » et un « LargeBlock », l'oiseau un « Greencar »... tout (ou presque) a été copié intégralement, sans rien modifier. J'ai l'impression d'avoir été floué d'autant plus que vous ne dites pas jusqu'à quel point se situe les emprunts. Je suppose donc que tout a été plagié. Non seulement le plagiat est un vol mais en plus en faisant cela vous ne progresserez jamais et resterez toujours incapable de créer des contenus originaux.
Compte-rendu : préprojet bien rédigé avec image : il s'agit de réaliser une copie d'un jeu connu T-Rex ; doc final très bien rédigé, je retiens la formule « seul on va plus vite mais à deux on va plus loin ». J'y croirai presque si je n'avais pas finalement compris que cela vous a mené jusqu'au Pérou...
Note : 05/20 (pour le travail de faussaire et les commentaires en français)

17.Investigate-the-Desk (escape game)
Utilisation : C'est un jeu difficile. J'y ai passé quelques heures à essayer de trouver l'ensemble des indices sans arriver au bout. Je ne suis pas familier de ce genre de jeu mais c'est amusant/intrigant/agaçant de s'y essayer ; je suppose que c'est gratifiant d'aboutir à la solution mais je n'aurai pas cette expérience (pas tout de suite en tout cas).
Programme : utilise PIL, tkinter, os. 600 lignes de code environ réparties sur 10 fichiers. Le niveau général de programmation y est élevé, largement au-dessus des attentes. Voilà par exemple, pour décoder un texte « encoded » encrypté avec un code César de clé « cle » :
"".join(chr(ord(i)-cle+26*((ord(i)-cle<65)-(ord(i)-cle>90))) if i!=' ' else ' ' for i in encoded).lower()
Ceci pour expliquer que je ne vais pas chercher à tout comprendre ni même critiquer ces lignes.
Compte-rendu : préprojet très avancé qui donne à peu près tout ce qui a été mis en place (et même les clés pour avancer dans le jeu, merci) ; doc final et programmes python sur internet, le doc est très bien rédigé, donne principalement le rôle des différents programmes, liste quelques problèmes et indique qu'une interaction avec le serveur permet de valider les codes trouvés (ainsi on ne les voit pas dans le programme). La note reflète l'impression générale de cet ensemble de programmes et également l'aide technique apportée à d'autres groupes.
Note : 20/20

18.Tablette Smartphone
Utilisation : Je n'ai pas pu tester le programme ; j'ai vu fonctionner certaines application mais l'ensemble n'a pas été remis.
Programme : Je n'ai pas eu le programme
Compte-rendu : préprojet assez précis, rédigé en forme de lettre « Bonsoir Monsieur... patati patata » ; doc final non rendu
Note : 07/20 (pour ce que j'ai vu, pour ce qu'on m'a rendu la note serait plutôt 02/20)
Finalement, avec 15 jours de retard (en partie excusé du fait de l'ENT qui n'a bizarrement pas envoyé le message le 3/3), j'obtiens votre projet. Celui-ci tout d'abord ne fonctionne pas, il faut importer folium qui est demandé par le programme map (le choix de ce nom n'est pas heureux car map est un mot réservé en Python :***map()*** peut être utilisée pour exécuter une fonction à chaque élément d'une liste de données). Sinon le programme fonctionne bien ; il est assez amusant, les applications sont diversifiées et hétérogènes : certaines ouvrent de petites fenêtres, d'autres prennent tout l'écran. Certaines sont très bien d'autres moins : l'IMC n'est pas affichée ! Le calcul de la moyenne de clics par seconde semble mal calculée ; le calcul des intérêts d'un placement est trop limité (intérêts simples, sans capitalisation) ; il n'y a pas de score calculé ou affiché pour pierre-

feuille ciseaux ou le morpion ni le nombre de coups pour le juste prix ; les conversions sont affichées telles quelles (un binaire comme '0b1100' pourrait être affiché '1100' pareil pour les hexadécimaux, les points décimaux pourraient être changés en virgules)... Qui trop embrasse mal étreint pourrait être le nom de votre application. Dans l'état, les applications apparaissent plutôt comme des prétextes à peine ébauchés pour rendre le projet crédible ; elles ne sont pas à utiliser pour elles-mêmes. Je pense qu'il faudrait mettre l'effort sur uniformisation des fenêtres crées (couleur, taille, mode de fermeture). Je manque de temps pour regarder les programmes en détail : ils semblent relativement bien construits et commenté (commentaires simples sur une ligne) sauf que c'est parfois bien compliqué il me semble. L'application IMC par exemple contient 3 fenêtres (IMC, F1 et F2), alors qu'il pourrait ne s'agir que d'une fenêtre, les infos s'affichant dans la fenêtre et l'IMC dans la zone prévue... Le plus curieux c'est que votre convertisseur fait cela très bien, pourquoi ne pas reproduire le même type de fenêtre pour l'IMC ? Sans doute est-ce là un défaut du travail de groupe : les applications ont été développées séparemment les unes des autres, sans un regard globalisant...
En bref, le projet est consistant et intéressant (j'ai bien aimé la carte, le panneau d'accueil et l'effort pour diversifier les unités de mesure dans le convertisseur). Il mériterait d'avoir davantage travaillé les applications ; le compte rendu final est bien rédigé et contient des informations détaillées ; compte tenu du retard pour la réception, je modifie la note en lui ajoutant 9 points (le retard vous fait perdre 1 point).

19.Fond d'écran (pour trois OS de la famille linux)
Utilisation : Sur mon ordi de la Région avec Linux Mint, j'ai essayé les opérations préconisées sans succès : j'ai vu l'image, cliqué sur le bouton mais rien ne s'est installé. Je suis un peu décu car cela devait fonctionner.
Programme : fonctionne avec les bibliothèques tkinter, os, json, requests, datetime et PIL. 126 lignes de code pour s'adapter aux 3 OS susceptibles d'utiliser ce programme ; les commentaires de fin de ligne sont assez explicites mais le sujet choisi est trop technique (manipulation de fichiers protégés, connection avec un site et récupération d'une image, navigation dans l'arborescence des dossiers, reconnaissance d'une OS, ...) et son exécution trop dépendante de l'opérateur
Compte-rendu : préprojet succinct avec l'idée générale et les bibliotèques visées ; doc final assez bref aussi mais donnant le mode d'emploi du fonctionnement. Le projet est à mon avis trop technique et un des membres du groupe, dépassé par cette technicité, n'a pas pu s'entraîner à programmer suffisamment (le transfert de compétences étant rendu impossible par la technicité) ce qui est pourtant le but d'un projet. La note est finalement une côte mal taillée entre les aspects positifs et négatifs du projet et tient compte également de l'énergie déployée vers les autres groupes pour dépanner certaines situations.
Note : 17/20

20.Spong (Pong revisité)
Utilisation : Le programme n'est pas terminé. Il fonctionne jusqu'à un certain point : on réussit à déplacer les joueurs (des souris?) et même à mettre la balle en mouvement et aussi à la frapper de manière à modifier son mouvement ; mais je n'ai pas réussi à faire mieux que ça. Quand un joueur gagne (je n'ai pas vraiment compris ce qu'il faut faire : détruire l'autre joueur?) le programme s'arrête, on dirait qu'il plante, plus rien n'est possible, il faut relancer le programme... Bon, mais la fenêtre est bien construite, avec des images pour indiquer les touches à utiliser. Je regrette qu'il n'y ait pas un petit texte à lire pour savoir quoi faire au début. C'est en plus un jeu difficile à manipuler seul (il faudrait avoir une bonne indépendance des mains...)
Programme : près de 650 lignes de code réparties en 6 fichiers, un par classe (sauf main qui se contente de lancer le programme). Nécessite pygame, pytmx, pyscroll, random et time.
Compte-rendu : préprojet bien complet et riche, très ambitieux pour un premier projet avec beaucoup de fonctionnalités compliquées et aucune piste sérieuse pour les réaliser (copier un pong classique serait déjà un bon début) ; doc final intéressant montrant comment les attentes ont du être

revues à la baisse, comment les méthodes se sont affinées malgré la sensation d'échec que cette exploration sans boussole de l'univers des jeux vidéos peut faire naître.
Note : 15/20

21.Jeu du président
Utilisation : Le jeu fonctionne très bien, exception pour la taille de l'écran qui n'est pas correcte : sur mon ordi en Windows je ne vois qu'une partie (les cartes du joueur de droite sont invisibles). L'écran d'accueil est amusant avec ses 3 encarts publicitaires avec de vrais liens vers des sites. L'aide est appréciable (le noir ne se voit pas trop), les autres boutons ne sont pas inutiles. Les parties ont un rythme rapide mais pas instantanée : on assiste à la reflexion des bots... sur un plan pratique, les boutons « passer » et « terminer » (pourquoi pas « jouer »?) sont trop proches, je me suis trompé plusieurs fois. La sélection des cartes à jouer fonctionne très bien. Esthétiquement, les couleurs ne sont pas trop bien choisies je trouve, peut-être que ce serait mieux de réserver un espace pour les affichages afin qu'ils ne soient pas écrits sur la table de jeu. J'ai pu faire 3 parties (je n'ai pas fini « trou du cul » grâce au bot Alice qui a terminé sur un « 2 », merci Alice !). C'est sympa de jouer contre des bots qui portent les prénoms de personnes de la classe.
Programme : 1500 lignes dans un seul fichier et toutes les images de carte à l'endroit et tournées de 90° réparties dans 2 dossiers. De très nombreux et longs commentaires des 2 sortes. L'écran d'aide est écrit ligne par ligne (près de 150 lignes à lui tout seul) alors qu'une seule aurait suffit avec une image. C'est tout de même très proprement réalisé : les déclarations de bouton sur 11 lignes, interligne, commentaire et placement compris ! Il y a des répétitions qui auraient pu être évitées, c'est vrai aussi, beaucoup de variables globales (sont-elles vraiment nécessaires? Ne le sont que celles concernant des variables, pas des listes, modifiées dans la fonction) mais dans l'ensemble c'est assez propre et compréhensible. Avoir tenté et réussi l'implémentation d'une IA est également une belle réussite, mais si vous pensez pouvoir l'améliorer.
Compte-rendu : préprojet bien rédigé, l'idée est délimitée, assez ambitieuse et précise ; très long doc final avec, il est vrai, de nombreuses images copies d'écran, mais l'ensemble des difficultés rencontrées est évoqué et elles ont été nombreuses, ne serait-ce que la coordination du travail à 3, les errements entre l'utilisation de la console, de pygame pour finalement tout faire sur tkinter. Vous dites vous être fait aider (par Jeremy ou sur internet) et c'est tout à votre honneur de le reconnaître, il n'y a pas de mal à ça d'autant qu'on ne peut pas vous accuser de plagiat à cause de ces aides bien comprises.
Note : 19/20

22.CyberOne
Utilisation : Fonctionne très bien, sauf qu'au départ j'ai eu une erreur avec la fonction types_bloc(self, data:list[list[int]]) et le message d'erreur « 'type' object is not subscriptable ». J'ai remplacé cela par types_bloc(self, data) et ça a fonctionné. Je ne vois pas bien pourquoi vous annoncez quel est le type des variables, ce n'est pas nécessaire en Python (sans doute une habitude venant d'autres langages). Le jeu est fluide, un peu difficile pour moi (j'ai réussi à aller presqu'à la fin du 2ème niveau mais quand j'ai perdu et qu'il fallait recommencer ce niveau j'ai (momentanément) renoncé ; mon avancement : 2183 marches, 633 sauts, 70 morts, 1 kill). Je dois dire qu'il est très bien réalisé, cela fait penser un peu au jeu de Jules, mais je reconnais l'originalité des parcours et et des obstacles (pour l'instant je n'ai affronté que le vide et un vilain robot monotâche mais je suppose que de plus grands dangers me guettent). J'ai aussi fini par trouver (par hasard) le bouclier... Super bonne idée mais du coup, je dois arriver à coordonner mes 2 mains... On verra plus tard. J'apprécie également la possibilité de recommencer au dernier niveau raté (plutôt que de tout recommencer). Vous avez poussé la réalisation assez loin et encore, je n'ai pas tout vu, mais grâce à la possibilité de tricher (entrer un niveau réussi supérieur), j'ai pu tester un peu le niveau 3, nettement trop dur pour moi... J'en resterai là
Programme : près de 850 lignes réparties en 3 fichiers. Bibliothèques utilisées pygame, os et csv. Le tout est bien écrit, les commentaires sont peu nombreux mais existent. J'essaie d'y trouver

l'explication de la petite image jaune en haut à gauche qui ne se met pas tout le temps.
Compte-rendu : préprojet assez succinct, l'idée est énoncée ; le doc final est bien écrit et assez précis sur quelques difficultés (double saut, etc.)
Note : 20/20