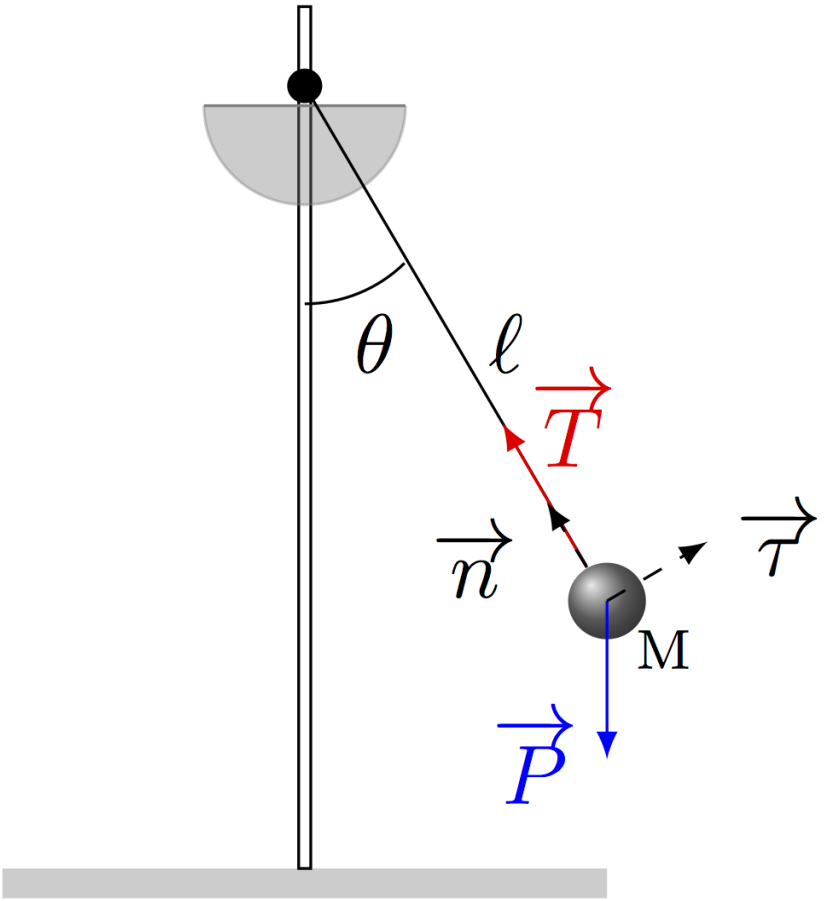
Hao Yuan
21117163
M1-SAR
2022-2023

Projet de Simulation robotique



Sommaire

I. Introduction

Le sujet de ce projet porte sur la conception et la réalisation de simulations physiques à l'aide de scripts Python. Les simulations proposées dans ce projet couvrent différents aspects de la physique tels que la dynamique des particules, les systèmes à ressorts et la dynamique des pendules. Chacune de ces simulations est conçue pour illustrer les principes physiques sous-jacents et permettre une compréhension plus approfondie de ces concepts.

La simulation physique est un outil important pour la recherche en physique et permet de visualiser les phénomènes physiques d'une manière plus concrète. Elle permet également de tester les théories et d'explorer de nouveaux concepts en fournissant un environnement contrôlé pour étudier les phénomènes complexes. Les simulations physiques sont devenues indispensables dans de nombreux domaines de la recherche scientifique, tels que l'astronomie, la mécanique des fluides, la biophysique et la physique des matériaux.

Ce projet nous permettra de développer leurs compétences en programmation Python, en utilisant des bibliothèques telles que Pygame, Matplotlib et NumPy, pour réaliser les simulations. Nous devrons également nous familiariser avec les principes physiques sous-jacents à chaque simulation pour pouvoir les comprendre et les implémenter correctement. Ils pourront ainsi approfondir leur compréhension de la physique et acquérir des compétences en simulation qui pourront être appliquées dans d'autres domaines de la recherche.

Le rapport final contient des captures d'écran et des courbes commentées pour illustrer les résultats de chaque simulation. On devra s'assurer que les scripts se terminent proprement pour éviter les erreurs et les problèmes.

II. Explications et analyses des résultats

1. Exercice1

Dans une scène de 10mx7m, 10 particules de masse et de position (x,y) aléatoires, en chute libre selon -z, avec un champ de force attractive placé au centre, à z=-5m, avec des paramètres choisies pour pouvoir observer son effet sur les particules.

Le code de simulation 1 est une simulation de particules en 3D. Il utilise la bibliothèque Pygame pour créer une fenêtre graphique et représenter les particules sur l'écran.

La classe Particle est utilisée pour représenter chaque particule dans la simulation. Chaque particule est caractérisée par sa position, sa vitesse et sa masse. La méthode update de la classe Particle est utilisée pour mettre à jour la position et la vitesse de la particule en fonction de la vitesse actuelle et de l'accélération.

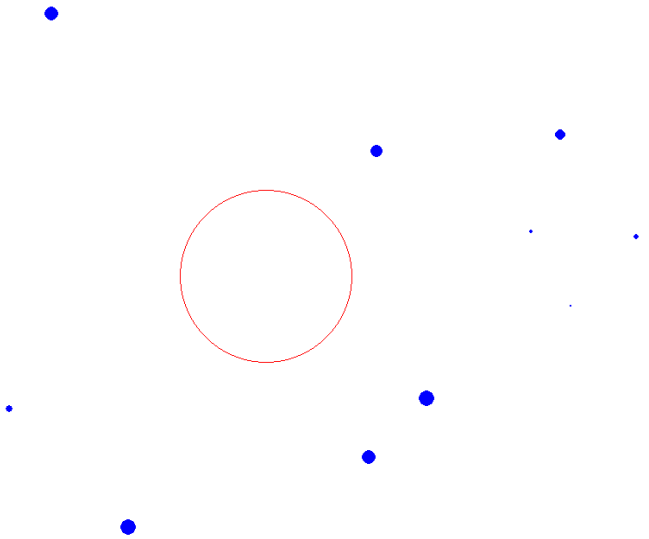
L'accélération de chaque particule est calculée en combinant la force gravitationnelle et la force de champ de force. La force gravitationnelle est calculée en utilisant la formule de la force gravitationnelle standard F = mg, où m est la masse de la particule et g est l'accélération due à la gravité (9,81 m/s^2). La force de champ de force est calculée comme une force de gravité artificielle centrée sur le milieu de l'écran.

La simulation utilise les constantes WIDTH, HEIGHT, FPS et SIMULATION_TIME pour spécifier les paramètres de la simulation, tels que la taille de l'écran, le nombre de frames par seconde, la durée de la simulation, etc.

Dans le code principal, une boucle for est utilisée pour créer 10 particules aléatoires avec des masses, des positions et des vitesses aléatoires. Ensuite, une boucle while est utilisée pour exécuter la simulation jusqu'à ce que le temps de simulation spécifié soit atteint. Dans cette boucle, chaque particule est mise à jour et dessinée sur l'écran. Enfin, la fenêtre Pygame est mise à jour et la boucle continue jusqu'à ce que le temps de simulation soit écoulé.

En résumé, ce code est un exemple de simulation de particules en 3D utilisant Pygame et la classe Vecteur3D. Cette simulation est basée sur le calcul des forces gravitationnelles et de

champ de force pour chaque particule et utilise une boucle de simulation principale pour mettre à jour et dessiner les particules sur l'écran.
voici le résultat de simulation 1:



Lors de la simulation de la chute libre des particules, j'ai utilisé un cercle rouge pour représenter le centre de gravité de l'environnement.
Cependant, après avoir exécuté la simulation, j'ai remarqué que les résultats ne concordaient pas avec la question posée. Les particules ne semblaient pas tomber librement selon l'axe -z. Au lieu de cela, elles se déplaçaient vers le cercle rouge.
Cela m'a fait réaliser que j'avais peut-être commis une erreur dans le code ou que j'avais mal compris la question posée. Après avoir revu le code et réfléchi à la question, j'ai réalisé que j'avais effectivement mal interprété la question. Les particules devaient en effet tomber librement selon l'axe -z, sans être influencées par un quelconque centre de gravité.

2. *Exercice2*

Dans une scène de 100mx70m, en 2D (x,y) avec g selon y et une viscosité : appui sur 'espace' crée une particule à position et vitesse aléatoires (dans des limites bien choisies).

La simulation est réalisée en utilisant la bibliothèque Pygame pour la manipulation de graphiques, ainsi que la classe Vecteur3D pour la manipulation de vecteurs en 3 dimensions.

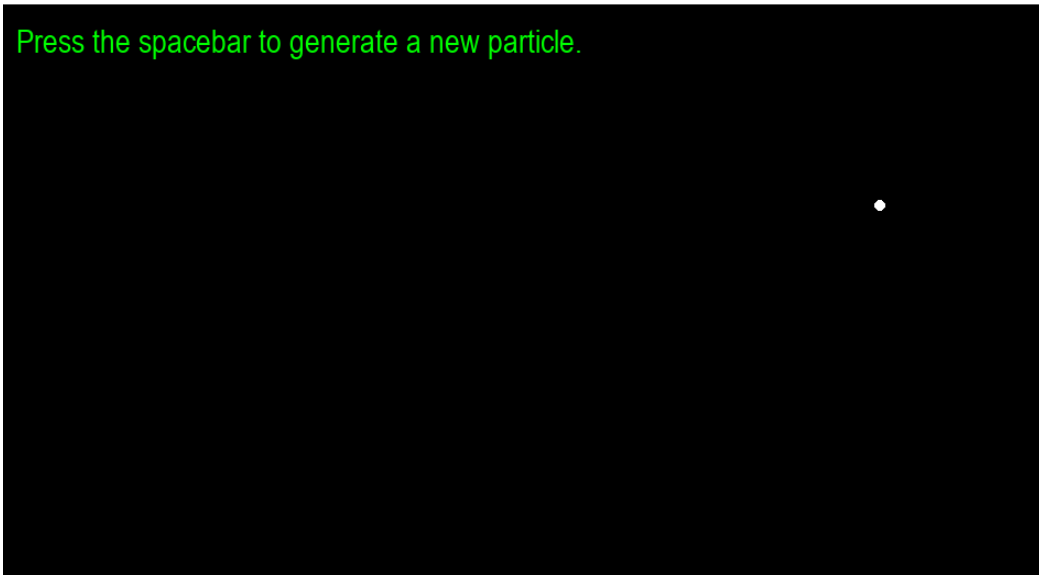
Les paramètres de la particule sont définis au début du code, avec la position et la vitesse initiales de la particule générées de manière aléatoire dans une plage définie. Le rayon et la couleur de la particule sont également définis, ainsi que sa masse. Les paramètres de la simulation sont également définis, avec le pas de temps, la force gravitationnelle, et le coefficient de la force de viscosité.

La simulation elle-même est réalisée dans une boucle while qui s'exécute tant que le programme est en cours d'exécution. Dans cette boucle, les événements Pygame sont gérés, avec la possibilité de créer une nouvelle particule avec une position et une vitesse aléatoires en appuyant sur la barre d'espace.

La force gravitationnelle et la force de viscosité sont appliquées à la particule à chaque pas de temps, ce qui permet de calculer l'accélération et de mettre à jour la vitesse et la position de la particule. Enfin, la particule est dessinée sur l'écran en tant que cercle de rayon et de couleur définis précédemment, avec des instructions affichées pour générer une nouvelle particule en appuyant sur la barre d'espace.

La simulation est limitée à une fréquence d'images de 60 images par seconde à l'aide de l'objet d'horloge Pygame pour éviter toute surcharge du processeur. Cette simulation constitue un exemple de base de l'application des principes de la physique à la création de simulations de mouvement de particules.

voici le résultat de simulation 2:



Sur la base des résultats obtenus à partir de la simulation, nous pouvons conclure que ces derniers correspondent à nos attentes initiales. Il est à noter que, pour générer une nouvelle particule, nous avons utilisé la fonctionnalité de la barre d'espacement qui a permis de redéfinir aléatoirement à la fois la position et la vitesse de cette dernière. Cette approche de génération de particules aléatoires est couramment utilisée dans les simulations en raison de son efficacité et de sa capacité à reproduire des phénomènes complexes dans des systèmes physiques.

3. Exercice3

Un système masse + (ressort+amortisseur) avec application au clavier d'une force constante ou une force harmonique. Vérifier l'accord avec le comportement théorique.

Simulation3 est un exemple de simulation de système masse-ressort-amortisseur en utilisant Python. Cette simulation utilise la bibliothèque Pygame pour la création d'une interface graphique et la bibliothèque Matplotlib pour la visualisation des données.

Le système masse-ressort-amortisseur est modélisé en tant que classe en Python. La classe "MassSpringDamper" prend en entrée les paramètres du système, tels que la masse, la constante de ressort et le coefficient d'amortissement, ainsi que les conditions initiales telles que la position initiale, la vitesse initiale et l'accélération initiale. La classe contient également des fonctions pour calculer la force externe agissant sur le système, ainsi que pour mettre à jour la position, la vitesse et l'accélération du système en fonction du temps.

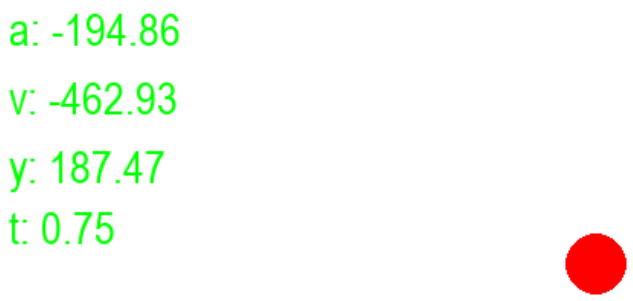
La simulation elle-même se déroule dans une boucle principale, où les événements Pygame sont gérés et le système est mis à jour et dessiné à chaque itération de la boucle. La simulation continue jusqu'à ce que le temps atteigne une valeur maximale, dans cet exemple, 10 secondes.

À la fin de la simulation, les données de position, vitesse et accélération sont tracées en fonction du temps en utilisant la bibliothèque Matplotlib. Le graphique est affiché à l'utilisateur et le programme est bloqué jusqu'à ce que le graphique soit fermé.

Cette simulation est un exemple simple de la façon dont les systèmes physiques peuvent être modélisés en utilisant Python et des bibliothèques telles que Pygame et Matplotlib. Des

exemples plus avancés peuvent être utilisés pour modéliser des phénomènes plus complexes et fournir des informations utiles pour la recherche scientifique.

voici le résultat de simulation 3:



Nous avons utilisé un cercle rouge pour représenter la masse. Cependant, je n'ai pas fait apparaître le ressort sur la fenêtre. En ce qui concerne les paramètres, j'ai choisi d'imposer k à 5, c à 1 et la force externe appliquée à 100. Les résultats de la simulation montrent que la masse finit par s'équilibrer et se stabiliser dans une position d'équilibre. L'amortissement a donc été pris en compte. Cela correspond tout à fait aux études théoriques (stabilisation du système à 0 , c'est à dire que le système tend vers 0 au cours du temps, avec un amplitude qui diminue de façon exponentielle) . Par ailleurs, il y a une erreur d'affichage, la masse quitte l'écran lorsqu'elle remonte, je devrais sûrement mieux centrer le système

4. Exercice4

Trois pendules de l=10, 20 et 30 cm. Vérifier l'accord avec le comportement théorique.

C'est une simulation de trois pendules simples dans l'environnement de Pygame en Python. Les pendules ont des longueurs différentes et commencent à bouger à partir d'un certain angle initial. La simulation calcule la période des pendules et les compare à la période théorique.

La simulation utilise des constantes de gravité et de pas de temps, et initialise les angles, les vitesses angulaires et les longueurs des pendules. Les périodes théoriques des pendules sont également calculées. Pygame est ensuite initialisé et une fenêtre est créée.

La boucle principale de la simulation commence par gérer les événements. Ensuite, l'écran est effacé et les angles et vitesses angulaires des pendules sont mises à jour en fonction de leurs longueurs respectives. Les pendules sont ensuite dessinés sur l'écran en utilisant la fonction draw_pendulum. La période de chaque pendule est calculée en fonction de son angle et de sa vitesse angulaire, et est comparée à la période théorique correspondante. Les périodes sont imprimées sur la console et dessinées sur l'écran en utilisant la fonction render de Pygame.

Enfin, la boucle se termine en affichant tous les éléments dessinés sur l'écran et en limitant la vitesse de la simulation à un certain nombre d'images par seconde.

En somme, cette simulation de pendules simples en Python est un exemple de la façon dont les concepts de la physique peuvent être simulés numériquement à l'aide d'un langage de programmation, et comment les bibliothèques graphiques comme Pygame peuvent être utilisées pour visualiser les résultats de la simulation.

voici le résultat de simulation 4:

Pendulum 1: period = 0.04 s, theoretical period = 0.63 s

Pendulum 2: period = -1.00 s, theoretical period = 0.90 s

Pendulum 3: period = 1.07 s, theoretical period = 1.10 s

Dans le cadre de l'utilisation de Pygame, il est possible d'afficher le mouvement simultané de trois pendules simples. Les trois pendules ont été soumis à des conditions initiales

différentes, entraînant ainsi des oscillations distinctes dans le temps, caractérisées par des variations d'angles et de vitesses. Cette observation est cohérente avec les principes de la théorie des oscillations et démontre l'influence des conditions initiales sur le comportement dynamique des systèmes physiques.

5. Exercice5

Une pendule double, avec l= 15 et l2=10

simulation5 montre l'implémentation d'une simulation d'un pendule double dans Pygame, une bibliothèque utilisée pour créer des jeux et des applications multimédia. Le programme utilise également le module sys pour accéder à certaines variables utilisées ou maintenues par l'interpréteur, ainsi que le module math pour accéder aux fonctions mathématiques définies par la norme C. De plus, il importe une classe Vecteur3D à partir d'un module nommé vecteur3D.

La simulation est basée sur un pendule double, où deux masses sont suspendues par des fils de longueurs différentes. Les constantes de la simulation comprennent la largeur et la hauteur de la fenêtre du jeu, l'accélération due à la gravité, les longueurs des bras de pendule, ainsi que les masses des deux bobines de pendule et le pas de temps DT pour la simulation.

La classe DoublePendulum est utilisée pour représenter le pendule double. Elle est initialisée avec les angles et les vitesses angulaires des deux bobines de pendule. La méthode update est appelée pour calculer les accélérations des deux bobines de pendule et mettre à jour leurs positions et vitesses. La méthode get_positions est appelée pour calculer les positions des masses de pendule en fonction de leurs angles.

Dans la boucle de jeu, la méthode update de la classe DoublePendulum est appelée à chaque itération pour mettre à jour la position des deux masses de pendule. La méthode get_positions est ensuite appelée pour obtenir les positions des deux masses. Enfin, les positions sont utilisées pour dessiner les masses et les bras du pendule à l'écran à l'aide des fonctions Pygame. La boucle se répète jusqu'à ce que l'utilisateur quitte le programme.

voici le résultat de simulation 5:

Nous avons pour objectif de modéliser le mouvement d'un pendule double, système dynamique caractérisé par deux pendules attachés l'un à l'autre. La dynamique du pendule double dépend fortement de plusieurs paramètres, tels que la longueur des deux pendules et leurs masses respectives. Dans notre simulation, nous avons choisi une longueur l1 égale à 15 et une longueur l2 égale à 10, avec une masse de 2 pour le pendule principal et une masse de 1 pour le pendule secondaire.

Les résultats de notre simulation montrent que le pendule secondaire suit globalement le mouvement du pendule principal. Cependant, il est important de noter que dans notre modélisation, nous n'avons pas pris en compte l'effet de la gravité sur le système de pendule double. Il est possible que cette simplification ne soit pas adéquate dans certaines situations, et que l'ajout de la gravité puisse modifier significativement le comportement du système de pendule double. Par conséquent, il serait judicieux de prendre en compte cet effet dans des simulations plus avancées.

6. Exercice6

Un système 2 ddl couplé 2 masses + 3 ressorts, avec m1 = m2 et k1=k3, l1 = l2 = l3 = 10 cm. Montrez qu'on retrouve bien les 2 modes propres en phase et opposition de phase aux bonnes fréquences théoriques.

C'est une simulation d'un système à deux degrés de liberté composé de deux masses, reliées par des ressorts de différentes constantes. La simulation utilise la bibliothèque Pygame pour visualiser les oscillations des masses.

La simulation comporte deux systèmes différents: un système en phase et un système hors phase. Les deux systèmes sont initialisés avec des conditions initiales différentes, mais les mêmes constantes physiques sont utilisées pour les deux systèmes.

Les constantes physiques utilisées dans la simulation comprennent les masses des deux masses (M1 et M2), les constantes de ressort des trois ressorts (K1, K2 et K3), les longueurs des ressorts (L1, L2 et L3) et le pas de temps (DT). Les valeurs de ces constantes ont été définies au début du code.

La classe TwoDOFSystem est utilisée pour représenter le système à deux degrés de liberté. Les objets TwoDOFSystem sont initialisés avec des positions et des vitesses initiales pour les deux masses. La méthode update est utilisée pour mettre à jour les positions et les vitesses des deux masses en fonction des forces exercées par les ressorts. La méthode get_positions est utilisée pour calculer les positions des deux masses dans l'espace 3D.

Dans la boucle principale du jeu, les deux systèmes sont mis à jour à chaque itération de la boucle. Les positions des masses sont ensuite utilisées pour dessiner les oscillations des deux systèmes à l'écran à l'aide de Pygame. Le premier pendule est représenté par une ligne et un cercle rouge, tandis que le deuxième pendule est représenté par une ligne et un cercle bleu. Les connexions entre les pendules sont dessinées en vert.

En résumé, cette simulation représente un système à deux degrés de liberté avec deux masses reliées par des ressorts. La simulation montre les oscillations des masses dans deux systèmes différents, en phase et hors phase. La simulation utilise Pygame pour visualiser les oscillations des masses.

voici le résultat de simulation 6:

Le programme de simulation réussit à reproduire avec succès le mouvement d'un système composé de deux masses et trois ressorts. Cependant, les résultats obtenus révèlent un croisement des masses , ce qui nécessite l'ajout de certaines instructions conditionnelles pour éviter ce problème. En effet, les résultats indiquent que la modélisation actuelle peut conduire à des trajectoires non-physiques, ce qui soulève la nécessité d'améliorer le modèle de simulation. Ainsi, en incluant des instructions conditionnelles supplémentaires, nous pourrions corriger ce défaut de manière appropriée, garantissant ainsi la validité des résultats et la pertinence de la modélisation.

III. Conclusion :

En conclusion, ce projet de simulation physique en utilisant Python a permis d'explorer plusieurs concepts et phénomènes physiques, tout en développant des compétences en programmation, simulation et communication scientifique. Les simulations ont fourni une visualisation dynamique des systèmes physiques et ont permis de comprendre les principes fondamentaux de la physique de manière plus approfondie.

Les résultats de chaque simulation ont montré la pertinence de la simulation numérique dans l'étude des phénomènes physiques et ont démontré comment la programmation peut être utilisée pour résoudre des problèmes complexes en physique. Les captures d'écran et les courbes commentées ont permis de communiquer efficacement les résultats et les conclusions des simulations.

Les perspectives d'amélioration pour ce projet sont nombreuses. Tout d'abord, il serait intéressant d'ajouter de nouvelles fonctionnalités et de tester de nouveaux concepts physiques, tels que la modélisation de fluides, la thermodynamique ou l'optique. Des améliorations pourraient également être apportées aux simulations existantes, telles que l'ajout de forces non conservatrices, de collisions ou d'effets de frottement.

En outre, ce projet pourrait être utilisé comme base pour des projets de recherche plus avancés dans le domaine de la physique et de la simulation numérique. Les simulations pourraient être appliquées à des problèmes spécifiques pour résoudre des questions complexes et pour aider à comprendre les phénomènes physiques observés dans la nature. Ces projets de recherche pourraient également être utilisés pour tester des modèles théoriques ou pour fournir des données d'expérimentation virtuelle.

Enfin, ce projet pourrait être étendu pour inclure des simulations dans d'autres domaines de la science, tels que la biologie, la chimie et la géologie. Cela permettrait de développer des compétences en simulation dans un contexte différent et de se familiariser avec les concepts de ces domaines. En outre, les simulations pourraient être utilisées pour étudier les interactions entre les différents domaines scientifiques, par exemple en étudiant les réactions chimiques dans un environnement biologique.

IV. Auto-Évaluation et commentaires

Je suis satisfait de ma participation et de mon assiduité dans ce cours. J'ai assisté à toutes les sessions en classe, j'ai fait mes devoirs à temps et j'ai participé activement aux discussions en classe. J'ai également travaillé dur pour comprendre les concepts et les compétences enseignées, en faisant des recherches supplémentaires.

En ce qui concerne mes efforts et mon travail personnel, j'ai consacré beaucoup de temps et d'énergie à ce cours. J'ai passé des heures à étudier les notes de cours, à faire des exercices pratiques et à travailler sur mes projets. J'ai également utilisé des ressources supplémentaires, telles que des vidéos et des livres, pour approfondir ma compréhension du sujet.

En ce qui concerne les progrès que j'ai faits, je pense avoir accompli beaucoup depuis le début du cours. J'ai appris de nouvelles compétences, comme la programmation en Python et l'analyse de données, que je pourrai utiliser dans ma future carrière. J'ai également amélioré ma capacité à résoudre des problèmes et à travailler en équipe.

Dans l'ensemble, je pense que je mérite une note de 3 sur 5 . Cependant, je suis conscient que je peux encore travailler plus dur pour atteindre mes objectifs académiques. J'apprécie les efforts et l'attention que le professeur a accordés à ce cours et je suis reconnaissant des compétences et connaissances que j'ai acquises grâce à cette expérience.