main.py

Update		
Entrée	**Traitement**	**Sortie**
	Invoquer les fonctions d'Update du player et de Mettaton.	

montreJeu		
Entrée	**Traitement**	**Sortie**
	Enlever sprNoir et txtControls Commence le dialogue de Mettaton	

commenceJeu		
Entrée	**Traitement**	**Sortie**
	Mettre le player sur le UI Jouer la musique Montrer le text du UI	

modules/setup.py

confirmation		
Entrée	**Traitement**	**Sortie**
L'input de l'utilisateur	Déterminer leur réponse	Retourner soit True ou False Print un message

setup		
Entrée	**Traitement**	**Sortie**
	Pour chacune des modules nécessaires, essaye d'importer. S'il y a une erreur, demande confirmation de l'utilisateur et installer.	

modules/tuteur.py

RandOperator		
Entrée	**Traitement**	**Sortie**
	Générer un opérateur mathématique aléatoire	Retourner le résultat

NombresCompatibles		
Entrée	**Traitement**	**Sortie**
Un opérateur mathématique	Selon quelques règlements simples, générer deux nombres compatibles	Retourner les résultats

FormatNombre		
Entrée	**Traitement**	**Sortie**
Un nombre	Transformer des nombres en string qui suivent un format désirable	Retourner le résultat

StyliseExpression		
Entrée	**Traitement**	**Sortie**
Une expression mathématique	Remplacer des exposants par des caractères spéciaux	Retourner le résultat

RandomExpression		
Entrée	**Traitement**	**Sortie**
	Créer une expression mathématique, en forme simple et stylisée.	Retourner les résultats

compileExpression		
Entrée	**Traitement**	**Sortie**
Le mode	Compiler une expression selon la difficulté d'entrée	Retourner les résultats

modules/jeu/arena.py

Arena.__init__		
Entrée	**Traitement**	**Sortie**
Le fenêtre Le canvas	Garder les entrées dans des variables Créer un rectangle	Retourner l'instance de l'objet

Arena.resize		
Entrée	**Traitement**	**Sortie**
Une largeur Une longueur	Changer la largeur et la longueur du rectangle	

Arena.move		
Entrée	**Traitement**	**Sortie**
Déplacement horizontal Déplacement vertical	Bouger le rectangle par un montant spécifié	

Arena.moveTo		
Entrée	**Traitement**	**Sortie**
Position horizontale Position verticale	Mettre le rectangle sur une position exacte	

modules/jeu/dialogue.py

Dialogue.__init__		
Entrée	**Traitement**	**Sortie**
Le fenêtre Le canvas Du texte Positions (x et y) Couleur S'il y aura une bulle ou non Les sons Le côté de la bulle S'il doit accepter l'input de l'utilisateur Une fonction à exécuter au fin	Garder les variables nécessaires Créer la bulle et le texte selon les paramètres Assigner des inputs si permis Préparer le texte	Retourner l'instance de l'objet

Dialogue.prepStr		
Entrée	**Traitement**	**Sortie**
Un string de texte	Assurer que le text de l'objet est “” Garder le string dans une variable Split le string en caractères, garder ces caractères dans une liste Créer un index pour les caractères Commencer à taper le texte	

Dialogue.printText		
Entrée	**Traitement**	**Sortie**
	Ajouter le prochain caractère au texte visible S'il y a de la ponctuation, ajoute une pause Jouer un son à chaque deux caractères Attendre un peu et recommence	

Dialogue.showText		
Entrée	**Traitement**	**Sortie**
	Arrêter Dialogue.printText et montre tout le text	

Dialogue.tryProgress		
Entrée	**Traitement**	**Sortie**
	Vérifier que Dialogue.printText est fini Si qu'on n'est pas au dernier string de texte, avance au prochain Autrement, détruire l'objet	

Dialogue.destroy		
Entrée	**Traitement**	**Sortie**
	Détruire la bulle et le text Enlève les keybinds Exécuter la fonction stockée	

modules/jeu/mettaton.py

MTT.__init__		
Entrée	**Traitement**	**Sortie**
Le fenêtre Le canvas Le player	Garder les entrées comme variables Créer le corps et les bras Initialiser plusieurs variables Initialiser le confetti	Retourner l'instance de l'objet

MTT.setAnim		
Entrée	**Traitement**	**Sortie**
Une liste de fichiers	Changer l'animation des bras	

MTT.message		
Entrée	**Traitement**	**Sortie**
Du texte Un côté Si on doit prendre en considération des inputs Une fonction à exécuter au fin	Créer un objet de Dialogue selon des paramètres précalculés et garder cet objet dans une variable	

MTT.move		
Entrée	**Traitement**	**Sortie**
Soit 1 ou 2	Commence à déplacer Mettaton entre deux positions possibles	

MTT.laser		
Entrée	**Traitement**	**Sortie**
Une fonction à exécuter au fin	Jouer un son Créer le laser et le blast Prépare pour MTT.unlaser	

MTT.unlaser		
Entrée	**Traitement**	**Sortie**
Une fonction à exécuter au fin	Endommager le player Après un moment, enlève le laser et le blast Exécute la fonction du fin	

MTT.Update		
Entrée	**Traitement**	**Sortie**
	Effectuer plusieurs changements périodiques : • Positionner le laser et le blast • Positionner le corps et le bras • Animer le corps et le bras	

modules/jeu/player.py

gameOver		
Entrée	**Traitement**	**Sortie**
Le player	Arrêter la musique et d'autres processus conditionnels Cacher le jeu Briser le cœur Jouer quelques sons Briser le cœur en plusieurs projectiles qui volent en dehors de l'écran Commence à afficher les résultats du jeu	

Player.__init__		
Entrée	**Traitement**	**Sortie**
Le fenêtre Le canvas L'aréna	Garder les entrées dans des variables Créer un Sprite d'un cœur rouge Initialiser plusieurs variables Créer les objets associés aux statistiques du player	Retourner l'instance de l'objet

Player.hurt		
Entrée	**Traitement**	**Sortie**
Un nombre Si on veut ignorer l'invincibilité Si on veut vibrer l'écran L'intensité de la vibration de l'écran	Déterminer si le player peut être endommagé S'il y aura encore plus que 0 HP, exécute Player.setHP, commencer l'invincibilité, jouer un son, et shake l'écran Si le HP va atteindre 0 ou moins, invoquer un gameOver	

Player.heal		
Entrée	**Traitement**	**Sortie**
Un nombre Si on veut dépasser la limite d'HP	Exécute Player.setHP Jouer un son	

Player.setHP		
Entrée	**Traitement**	**Sortie**
Un nombre	Changer la variable d'HP Repositionner et changer le HP bar et le texte d'HP	

Player.setMaxHP		
Entrée	**Traitement**	**Sortie**
Un nombre	Changer la variable de MaxHP Repositionner et changer le MaxHP bar et le texte d'HP	

Player.setLV		
Entrée	**Traitement**	**Sortie**
Un nombre	Changer la variable de LV Si on augmente le LV, jouer un son Calculer le nouveau MaxHP et Player.setMaxHP Changer l'indicateur de LV	

Player.setDir		
Entrée	**Traitement**	**Sortie**
Un nombre Un boolean	Changer la direction dans la liste de directions à être soit actif ou inactif (mouvement)	

Player.Update		
Entrée	**Traitement**	**Sortie**
	Contrôler le mouvement dans l'aréna Garder le player dans l'aréna Garder le player sur le niveau maximal du canvas Vérifier pour la collision Contrôler l'invincibilité causé par Player.hurt	

Player.bindInput		
Entrée	**Traitement**	**Sortie**
Le mode d'input (string)	Changer les inputs directionnels	

modules/jeu/question.py

Pointage.__init__		
Entrée	**Traitement**	**Sortie**
	Créer un objet qui garde en compte les points	Retourne l'instance de l'objet

Pointage.add		
Entrée	**Traitement**	**Sortie**
Un nombre	Ajoute le nombre aux points	

Option.__init__		
Entrée	**Traitement**	**Sortie**
Le canvas L'objet de Question Du texte Quel bouton à utiliser Quelle position à utiliser Si l'option représente le correct réponse	Garder les valeurs nécessaires dans des variables Créer un bouton Créer du texte	Retourner l'instance de l'objet

Option.reponse		
Entrée	**Traitement**	**Sortie**
Si l'option est le correct réponse ou non	Si l'option est correcte, montrer le confetti, montrer un message, changer le texte à vert, Player.heal, jouer un son, ajouter un point, parfois ajoute un LV au player, et mettre tout à l'état de repos. Si l'option est incorrecte (ou il n'y avait aucun réponse), changer le texte à rouge, affiche un message, montrer le correct réponse en vert, commence MTT.laser Enlever les boutons Changer des couleurs Ajouter chaque option et le couleur de son texte à la liste de résultats Ajouter le temps à la liste de temps Enlever le timer Enlever la question Unbind le clé 'c'	

Question.__init__		
Entrée	**Traitement**	**Sortie**
Le fenêtre Le canvas Le player Mettaton Le UI Du texte Une liste de réponses Un nombre qui représente le correct réponse Le temps	Garder les valeurs nécessaires dans des variables Créer le texte de la question Faire ce texte vibrer Ajouter la question à la liste des questions Créer quatre options Créer la notification d'indice (et bind 'c' à Question.hint) Créer et commencer un timer	Retourner l'instance de l'objet

Question.hint		
Entrée	**Traitement**	**Sortie**
	Sélectionner deux nombres entre 0 et 3 qui ne correspondent pas à la correct réponse Changer les options correspondantes pour être gris et enlève leur habileté d'être sélectionné Jouer un son Player.hurt Enlever le texte d'indice Unbind le clé 'c'	

Question.updateTimer		
Entrée	**Traitement**	**Sortie**
	Enlever 1 unité du timer à chaque seconde Si le timer arrive à 0, traite-la comme une mauvaise réponse	

modules/jeu/sprites.py

Sprite.__init__		
Entrée	**Traitement**	**Sortie**
Le canvas Une image Deux nombres (position) SI on veut ajouter des pixels transparents aux frontières L'anchor	Initier des variables Garder les valeurs nécessaires dans des variables Créer l'image Ajouter l'image à une liste globale pour garder le sprite sur l'écran	Retourner l'instance de l'objet

Sprite.adjust		
Entrée	**Traitement**	**Sortie**
	Ajouter des pixels transparents à l'image du sprite	Retourner l'image générée

Sprite.move		
Entrée	**Traitement**	**Sortie**
Deux nombres	Bouger le sprite par quelques pixels	

Sprite.moveTo		
Entrée	**Traitement**	**Sortie**
Deux nombres	Mettre le sprite à une position exacte	

Sprite.set		
Entrée	**Traitement**	**Sortie**
Une image Si on veut exécuter Sprite.adjust sur l'image Si on veut remplacer l'image de base	Adjust si nécessaire Changer des variables Changer l'image de l'objet	

Sprite.color		
Entrée	**Traitement**	**Sortie**
Une couleur Une couleur à remplacer	Si aucune couleur est spécifiée pour être remplacé, utilise la couleur courant du sprite Pour chaque pixel de l'image du sprite qui match celle à remplacer, remplace-la avec la nouvelle couleur Remplacer l'image du sprite avec la nouvelle	

Sprite.rotate		
Entrée	**Traitement**	**Sortie**
Une rotation, en degrés	Remplacer l'image du sprite avec une qui a subi une rotation	

Sprite.scale		
Entrée	**Traitement**	**Sortie**
Deux nombres Si on veut remplacer l'image de base du sprite	Multiplier les dimensions de l'image, remplacer l'image du sprite	

Collider.__init__		
Entrée	**Traitement**	**Sortie**
Le canvas Une image Le player Deux nombres (position) Si on veut adjuster l'image Une fonction à exécuter Si cette fonction doit seulement exécuter une fois	Initier un Sprite Obtenir les positions des frontières du sprite Garder les valeurs nécessaires dans des variables Ajouter l'objet à la liste d'objets avec lequel le player peut rentrer en collision avec Garder la position de l'objet dans cette liste	Retourner l'instance de l'objet

Collider.getBorders		
Entrée	**Traitement**	**Sortie**
	Mettre à jour les coordonnées des frontières du sprite	

Collider.onCollide		
Entrée	**Traitement**	**Sortie**
	Si la collision peut seulement arriver une fois, prévenir une autre collision. Exécuter la fonction du Collider	

Collider.destroy		
Entrée	**Traitement**	**Sortie**
	Enlever le Collider de la liste dans le player Mettre à jour les indexe des autres Colliders Enlever le sprite du Collider	

Gif.__init__		
Entrée	**Traitement**	**Sortie**
Le fenêtre Le canvas Une image (.gif) Deux nombres (position) La vitesse du gif Combien de frames on veut essayer d'extraire Après combien de frames on veut diminuer l'opacité	Garder les valeurs nécessaires dans des variables Extraire les frames du fichier de gif, transformant l'opacité si nécessaire Initier un objet d'image	Retourner l'instance de l'objet

Gif.Play		
Entrée	**Traitement**	**Sortie**
Si on veut interrompre le loop courant et recommencer	Recommencer le gif si nécessaire Avancer d'un frame (changer l'image) Gif.Play encore après un peu de temps	

modules/jeu/ui.py

compileResults		
Entrée	**Traitement**	**Sortie**
Le UI Si on veut sauter l'étape de cacher tout	Vérifier qu'il y avait une question (si non, bouger le player au bouton de 'prochain') Cacher le player, désactiver tout mouvement Arrêter Mettaton et changer ses Sprites à gris Créer un Sprite noir qui couvre l'écran avec opacité 0 Si on ne veut pas sauter cette étape, augmente graduellement l'opacité du Sprite noir Jouer un son de drumroll Pour chaque question (en pages horizontales si nécessaires), créer des objets de texte qui montrent la question, les réponses (en couleur), et s'ils étaient bien Calculer le pourcentage de questions correct Jouer un son différent selon le pourcentage Manipuler des objets de UI pour montrer le montant de questions Montrer le pourcentage Montrer l'indicateur des pages Bind les touches directionnelles horizontales à une fonction qui change le page Calculer et montrer le temps moyen Montre des instructions de contrôles spécifique à cet écran Bind le clé d'échap pour fermer le fenêtre	

commenceQuestion		
Entrée	**Traitement**	**Sortie**
Le UI	Enlever le texte du menu Générer une question Générer trois fausses réponses à cette question Compiler ces réponses aléatoirement avec le correct réponse Bouger Mettaton et changer son animation Mettre le player dans l'aréna Compiler une objet de Question	

UI.__init__		
Entrée	**Traitement**	**Sortie**
Le fenetre Le canvas Le player Mettaton	Garder les entrées dans des variables Mettre une référence au UI dans le player Créer deux boutons Collider (PROCHAIN, QUITTER)	Retourner l'instance de l'objet

UI.menuText		
Entrée	**Traitement**	**Sortie**
Du texte	Générer un objet Dialogue avec des paramètres idéals Garder cet objet dans une variable	

UI.movePlayer		
Entrée	**Traitement**	**Sortie**
	Mettre le player sur le bouton de PROCHAIN et changer son input	

UI.hightlightBtn		
Entrée	**Traitement**	**Sortie**
Le bouton	Préparer à soit commenceQuestion ou compileResults Basée sur le bouton sélectionné, détermine quoi faire Changer le bouton à sa version jaune Reset l'autre bouton Jouer un son Bind le clé 'z' à une fonction qui joue un son et exécute les fonctions mentionnées ci-dessus Jouer un son	