**Accueil :**

**=> Lancer le jeu => chargement de la page Menu**

**=> Quitter le jeu => Fermeture du programme**

**=> Menu :**

**= > Lancer la partie :**

=> Renseigner avec quel pokemon jouer selon la liste pokemon.json \*

menu déroulant avec les noms des pokémons disponibles

créer un objet de la classe pokémon(joueur = pokemon(...))

=> Choix aléatoire du pokémon adverse selon la liste pokemon.json

ouvrir fichier pokemon.json as liste

créer un objet de la classe pokemon(adversaire = pokemon(...))

=> LVL adversaire :

Si LVL du joueur == 1 ou LVL joueur== 2, LVL adv compris entre LVL joueur et LVL joueur +2.  
 Si LVL joueur > 2, LVL adv compris entre LVL joueur -2 et LVL joueur +2.

=> ajuster LVL Adversaire =

- for i in range(1, niveau adv)

hp += random.randint(1, 4)

attack += random.randint(1, 4)

défense += random.randint(1, 4)

=> choix du terrain:

if adversaire.type == “eau”:

cercle = pyagme.image.load(“chemin d'accès 3.png”)

elif adversaire.type == “terre”:

cercle = pygame.image.load(“chemin d’accès 5.png”)

elif adversaire.type == “feu”:

cercle = pygame.image.load(“chemin d’accès 1.png)

elif adversaire.type == neutre:

cercle = pygame.image.load(“chemin d'accès 6.png)

=> Partie lancée

=> système de combat

=> Classe Combat

=> **Méthodes de la classe**  => **récupère** type adversaire et PA adversaire en suivant le tableau établi (fichier type.json)

calculer PA du joueur et stocker dans une variable joueurAttaqueCombat

=> **définir** qui attaque en premier :

jet de dés par opposition avec bonus d’initiative

if random.randint(1, 20)+joueur.initiative>random.randint(1, 20)+adv.initiative:

tour du joueur()

tour de l’adversaire()

else:

tour de l’adversaire()

tour du joueur()

=> **calculer** si l’attaque rate ou touche

jet de dés réussite en fonction de la précision

touche( joueur, adv):

if random.randint(1, 100) < 80-((80\*adv.defense)/100):

attaque(joueur, adv)

else:

animation attaque bruit raté

print(f“{joueur.nom} à raté son attaque”)

=> **enlève** PV en fonction de l’attaque

=> **check** si la vie du pokémon est <= 0

mort(pokémon):

if pokemon.PV < 0:

return True

=> **renvoie** le nom du vainqueur

if joueur.PV > 0:

le joueur a gagné

else:

l’adversaire a gagné

=> **renvoie** les noms des pokémons gagnants et/ou perdant

=> **enregistre** les pokémons rencontrés dans le fichier pokedex.json :

=> vérification des doublons (si doublons, mise à jour du nombre de rencontres)

=> enregistrement sous la forme (nom, type, def, PA, PV)

=> Gain LVL à définir

=> Si victoire, possibilité de relancer un combat aléatoire

=> Gestion du gain de LVL et de l’évolution

=> Si défaite, retour à la page Menu

**=> Ajouter un Pokémon dans le fichier pokemon.json :**

=> Carac des pokémons :

=> nom, type, PV, PA, DEF, initiative

=> vérifier que chaque valeur correspond soit à un str pour le nom et type soit int pour les autres carac. Vérifier que le type est parmi les types admis.

=> check si le pokémon existe déjà

=> message ajout du nouveau pokémon

**=> Accéder au Pokedex :**

=> Affichage du pokedex via le fichier pokedex.json: