

Dossier ISN

LE PENDU

Membres du groupe :

HOUSSENBAY Ammar Raoul

TS 1

RALISON Harifidy Milanto

TS 1

ERI YANG Ha nna Safidy

TS 2.

I Regles du jeu

II Cahier des charges

III Realisation.

IV Chronologie et evolution de l'interface

V Bilan et perspectives.

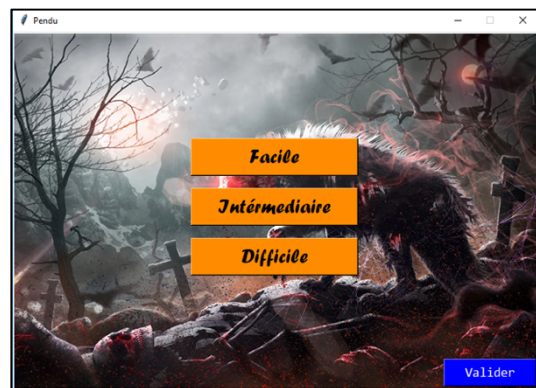
VI Annexes.

I] RÈGLES DU JEU :

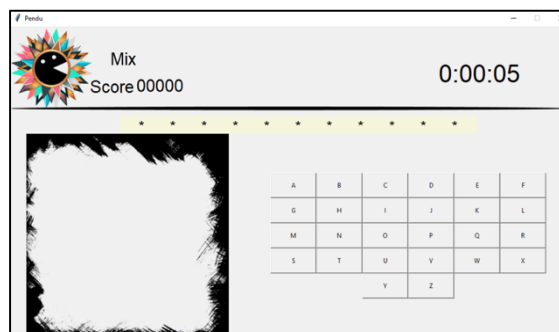
Le but de ce jeu est très simple : dans les règles classiques, un joueur choisit un mot qui existe et doit le faire deviner à un ou un ensemble d'autres joueurs. Pour ce faire, il écrira autant de tirets qu'il y a de lettres dans le mot. Les joueurs 'devineurs' proposent alors une lettre au hasard, et si celle-ci est dans le mot, la lettre est inscrite à sa place, sur le tiret, et autant de fois que la lettre apparaît dans le mot. Dans le cas échéant, si la lettre n'est pas dans le mot, le premier trait du pendu est dessiné. Il y a 10 traits à dessiner dans notre jeu afin de compléter le dessin du pendu, donc 10 tentatives pour essayer de deviner le mot.



Ensuite, le joueur choisit son niveau de difficulté : facile, intermédiaire ou difficile.



Enfin, le joueur essaie de deviner le mot en proposant des lettres, sachant qu'il a 10 chances. Un chronomètre est affiché en haut à droite, afin de lui informer du temps qu'il a mis à deviner ou à avoir essayé de deviner le mot en question, à titre indicatif.



II] CAHIER DES CHARGES :

a) Contexte et présentation du projet

Dans cette première sous-partie, il s'agira de décrire le contexte et la démarche suivie.

Contextuellement, notre but était de créer le jeu du pendu, virtuellement.

L'objectif pour nous n'était pas uniquement de présenter un programme qui fonctionne et qui est compréhensible, mais également de présenter une interface attractive, qui

donnerait envie de jouer puisqu'il s'agit avant tout d'un jeu visuel (les traits du pendu qui se dessinent progressivement sur l'écran, création de 3 fenêtres distinctes pour les 3 parties du projet etc.).

b) Besoins et contraintes liées au projet

Afin de mener à bien notre projet, il était essentiel pour nous d'effectuer une carte mentale (qui sera présentée dans l'annexe) afin de visualiser nos besoins et nos contraintes, mais aussi pour organiser nos idées. Nous avons tout d'abord besoin d'un modèle sur lequel nous baser pour les emplacements des fonctions dans les fenêtres, utiliser des documents ressources qui serviront de base pour notre futur programme. L'idée de rajouter des niveaux et le chronomètre nous étaient venus tout seul.

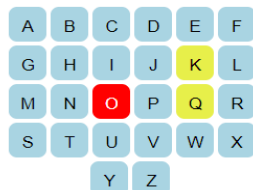
En terme de contraintes, nous avons programmé uniquement en Python (version 3).

LE PENDU EN LIGNE



Trouvez un mot de 8 lettres en 11 coups

- - - O - - - -
ERREUR : La lettre Q n'appartient pas au mot



Il vous reste 9 essai(s)

Source du jeu :

<https://www.funmeninges.com/jeu-du-pendu.html>

II | RÉPARTITION DES TÂCHES :

<u>Dates.</u>	<u>Planning.</u>	<u>Noms.</u>	<u>Tâches.</u>
9/01/2019.	<p>◇ Elaboration d'une carte mentale pour la structure des idées.</p> <p>◇ Division des tâches à <u>effectuer pour la séance prochaine.</u></p> <p>◇ Répartition des tâches.</p>	<p>Ammar.</p> <p>Harifidy.</p> <p>Safidy.</p>	<p>Focalisation sur le programme/script :</p> <ul style="list-style-type: none"> Fonction « niveau », fonction « choix d'un mot au hasard » (téléchargement d'un fichier texte) après choix du niveau par le joueur et fonction qui remplace nombre de lettres par des étoiles. Fonction qui vérifie si la lettre est à l'intérieur du mot à deviner. Fonction « nombre d'essais restants » et fonction « reessayer ».

16/01/2019.	<p>◇ Mise en commun et rassemblement des codes.</p> <p>◇ Idée du rajout du chronomètre.</p> <p>◇ Elaboration de l'interface Tkinter.</p>	<p>Ammar.</p> <p>Harifidy.</p> <p>Safidy.</p> <p>Tous les 3.</p> <p>Ammar et Harifidy.</p>	<p><u>Focalisation sur le programme/script :</u></p> <ul style="list-style-type: none"> • Recherche d'un programme pour le chronomètre. • Mise en commun des codes : travail en fonctions. • Organisation et structure de l'interface (combien de fenêtres...) <p><u>Focalisation sur l'interface :</u> Division des tâches à effectuer pour la séance prochaine en 3 :</p> <ul style="list-style-type: none"> • Programme pour la première fenêtre (titre, pseudo, boutons valider + erreurs éventuelles, et niveaux). • Programme pour la deuxième fenêtre (boutons A,B,C..., dessin du pendu, étoiles qui correspondent au mot et chrono).
23/01/2019.	<p>◇ Mise en commun et rassemblement des codes.</p>	<p>Ammar.</p> <p>Harifidy.</p> <p>Safidy.</p> <p>Tous les 3.</p> <p>Ammar et Safidy.</p>	<p>Peaufinage de nos codes respectifs :</p> <ul style="list-style-type: none"> • Résolution du problème d'attribution de commandes aux boutons pour la deuxième fenêtre. • Etudes des éventuelles erreurs (le joueur ne rentre pas de pseudo, re-choisit la même lettre...).
06/02/2019.	<p>◇ Idée de rajouter un score.</p> <p>◇ Idée de mettre une troisième fenêtre : mettre le choix des niveaux à part.</p> <p>◇ Idée de rajouter de la musique.</p>	<p>Ammar.</p> <p>Harifidy.</p> <p>Safidy.</p>	<ul style="list-style-type: none"> • Recherche d'un code pour mettre de la musique. • Résolution du problème d'attribution de commandes aux boutons pour la deuxième fenêtre.

			<ul style="list-style-type: none"> Recherche d'un code pour le score.
13/02/2019.	<ul style="list-style-type: none"> ◇ Mise en commun et rassemblement des codes. ◇ Idée de rajouter plusieurs avatars (animés si possibles). ◇ Bilan de ce qu'il reste à faire. 	<ul style="list-style-type: none"> Ammar. Harifidy. Safidy. Ammar et Harifidy. 	<ul style="list-style-type: none"> Résolution du problème d'attribution de commandes aux boutons pour la deuxième fenêtre. Recherche d'un code pour rajouter des animations. Recherches des avatars et comment rajouter de la musique. <p>Bilan de ce qu'il reste à faire :</p> <ul style="list-style-type: none"> Code pour l'élaboration d'un classement et du score. Recherche encore d'un code pour rajouter de la musique. Rassemblement des scripts. Création d'une base de données pour la sauvegarde des scores.

Pour rassembler nos codes, ou par exemple lorsque le travail de l'un dépendait du code de l'autre, nous utilisons le Drive où nous avons créé un dossier et aussi les réseaux sociaux (Messenger où nous avons créé un groupe).

III | RÉALISATION.

◆ ERI-YANG Ha-nna Safidy Safidy.

En ce qui me concerne, mon rôle a été d'élaborer une fonction pour vérifier que la lettre entrée par l'utilisateur était dans le mot, puis d'esquisser une première approche de notre première fenêtre (à l'époque où il n'avait que 2 fenêtres) et enfin de chercher un code pour mettre de la musique dans le fond.

Je présenterai ici 2 fonctions. Une fonction en rapport avec le fonctionnement du programme une autre en rapport avec l'interface.

Tout d'abord, pour ma première fonction La stratégie principale, avant même d'écrire le programme était de créer une liste_alphabet c'est-à-dire avec toutes les lettres de l'alphabet à l'intérieur, et d'enlever petit à petit les lettres déjà tapées de la liste d'alphabet par l'utilisateur afin de l'empêcher de re-taper une deuxième fois la même lettre. Il fallait ensuite utiliser la condition « if » (avec les deux conditions suivantes : si la lettre tapée est dans le mot à deviner et si la lettre tapée est dans la liste alphabet).

La difficulté pour moi durant toute la phase d'élaboration du programme pour le pendu était que le jeu lui-même demandait une approche plutôt visuelle (que nous n'avions pas encore) ainsi que le fait que nos codes dépendaient l'un de l'autre. Donc, par exemple, pour vérifier que la lettre était dans le mot à deviner, je devais choisir un mot avant.

En ce qui concerne la partie interface, j'avais donc, comme dit précédemment, fait l'esquisse de notre première fenêtre avec le titre, l'entrée d'un pseudo, le choix du niveau et le bouton « valider », tout en pensant à ce que pourrait être l'esthétique de notre jeu, l'emplacement des différents éléments ([cliquez ici pour voir la fenêtre](#)). J'avais pour cette partie, rencontré plusieurs difficultés : j'avais pensé à plusieurs erreurs comme l'utilisateur ne rentrait pas de pseudo ou il ne choisissait pas de niveau ou il n'entrait ni son pseudo ni son niveau...) qu'il fallait absolument intégrer dans le script. Déjà, je n'avais pas de place pour écrire une quelconque erreur comme par exemple « Entrez votre prénom d'abord » (et aussi peut-être que l'utilisateur ne souhaite pas mettre de pseudo/prénom ?), alors ma solution était de mettre le message d'erreur dans l'entrée destinée au pseudo.

Ensuite, ma dernière tâche était de rechercher un moyen de rajouter de la musique, qui était une idée vraiment intéressante pour dynamiser notre jeu, mais cette démarche n'a malheureusement pas abouti. En effet, par la même occasion j'ai découvert le module pygames qui permettait notamment d'importer de la musique, de la jouer, de faire pause etc. Le souci était qu'il n'était pas installé dans Python. Malgré plusieurs tentatives (installation d'Anaconda, installation de codes pip dans le Terminal...), il m'était finalement impossible d'avoir le module pygames.

◆ HOUSSENBAY Ammar Raoul.

Pour ma partie, je me suis occupé de réaliser le chronomètre, le dessin du pendu, l'attribution des scores, ainsi que l'esthétique globale de la fenêtre passant par la mise en place des successions de fenêtre, le choix des avatars ou le changement de couleurs lors d'une sélection.

L'une de mes premières préoccupations a été de trouver la manière de mettre de l'animation à l'aide de Tkinter. Suite à quelque recherche j'ai découvert l'existence de la fonction .after. Je l'ai utilisé sans tarder afin de permettre l'affichage d'une horloge dynamique. Une fois les différents problèmes d'affichage réglés, j'ai créé une fonction attribuant un score à la personne en fonction du temps qu'elle a mis et de la difficulté du niveau.

Un autre problème est survenu à ce moment-là, il m'était impossible de récupérer l'heure à laquelle l'utilisateur finissait le jeu et en conséquent lui donner son score uniquement à partir de l'horloge que j'ai créée. J'ai dû alors créer une autre variable qui prend comme valeur +1 à chaque seconde. Ainsi dès que l'utilisateur fini le jeu, je récupère le contenu de cette variable correspondant au temps en seconde que l'utilisateur a mis pour finir le jeu.

Une fois toutes les tâches principales finies, j'ai pensé à l'interaction entre l'utilisateur et le programme. En effet il était primordial que l'interface soit pensée de manière correcte afin d'éviter des erreurs, par exemple nous avons pensé au fait que l'utilisateur ne

saisissent aucun nom ou bien aucun niveau, nous avons aussi pensé au fait qu'un utilisateur appuie successivement sur un bouton provoquant alors un ralentissement et la fermeture du programme, dû à une saturation de mémoire.

Malheureusement l'une des plus grandes difficultés est survenue lors de la création du programme principale, en effet je pensais au début créer plusieurs script python auxquelles je ferais appel en fonction de la fenêtre concernée. Or il s'est avéré que cette solution est impossible, en effet il était impossible de récupérer des valeurs d'une fonction à une autre, d'une fenêtre à une autre, de changer les paramètres d'un bouton, son affichage.

J'ai alors trouvé comme solution d'utiliser les fonctions `.Toplevel()`, permettant de créer une nouvelle fenêtre et `.grab_set` ainsi que `focus_set` permettant de porter l'attention (clic et interaction) sur la nouvelle fenêtre. Puis il fallait masquer l'ancienne fenêtre afin de ne pas avoir une succession de fenêtre ouverte, d'où l'emploi de `.withdraw`.

◆ RALISON Harifidy Milanto.

Personnellement, je me suis occupé de créer les 3 listes de mots différentes extraites d'un fichier `.txt` selon le niveau sélectionné, je me suis occupé de tout ce qui concerne l'affichage graphique du mot à deviner et de la création de tous les boutons correspondants à chacune des lettres de l'alphabet, des fonctions permettant d'afficher les lettres sélectionnées dans le mot à deviner une fois que l'utilisateur clique sur le bouton correspondant à la lettre, de la création d'une base de données pour stocker les scores réalisés par les joueurs, d'établir un classement des joueurs en fonction de leur score et de l'afficher.

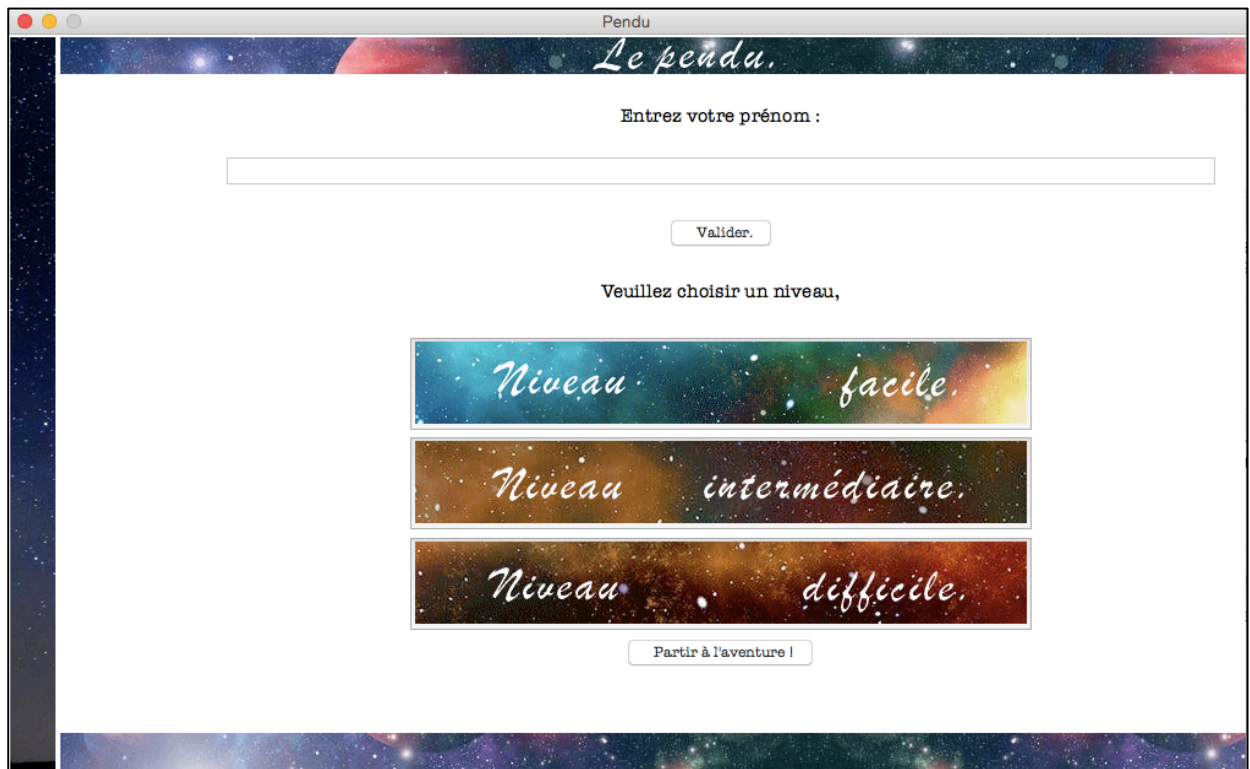
J'ai rencontré divers problèmes pour lesquels j'ai trouvé diverses solutions tels que l'attribution d'une commande avec un paramètre à un bouton que j'ai pu résoudre grâce à la découverte de la fonction `lambda`, la modification de boutons sans connaître leur nom en passant par le biais d'une liste pour pouvoir les modifier en fonction de leur rang, l'ajout d'un paramètre selon lequel l'état du bouton est désactivé pour régler le problème de son activation automatique.

Après avoir réussi à faire fonctionner en grande partie notre jeu du pendu, et l'attribution de score, nous avons eu l'idée de pouvoir sauvegarder les scores dans une base de données afin qu'un même utilisateur puisse y jouer et conserver son score anciennement enregistré plutôt que de devoir repartir à 0 à chaque fois. Je me suis donc occupé de cette partie, et j'ai fait la découverte du module `os.path` et du module `pickle` permettant respectivement d'appeler un fichier, ou vérifier qu'un fichier existe et d'ensuite le récupérer et donc de pouvoir exploiter les données s'y trouvant ; et la fonction `pickle.load()` et `pickle.dump()` permettant respectivement de récupérer tels quels les données d'un fichier et de déposer tels quels des données tels que un dictionnaire dans un fichier.

Après avoir réussi à réaliser un stockage des données, nous avons eu l'idée afin de les exploiter aux mieux, de réaliser un classement des meilleurs joueurs en fonction de leur score. J'ai donc réussi à établir un classement correct, cependant il renvoie des erreurs dans le cas où plusieurs personnes ont le même score.

IV] CHRONOLOGIE ET EVOLUTION DE NOTRE INTERFACE.

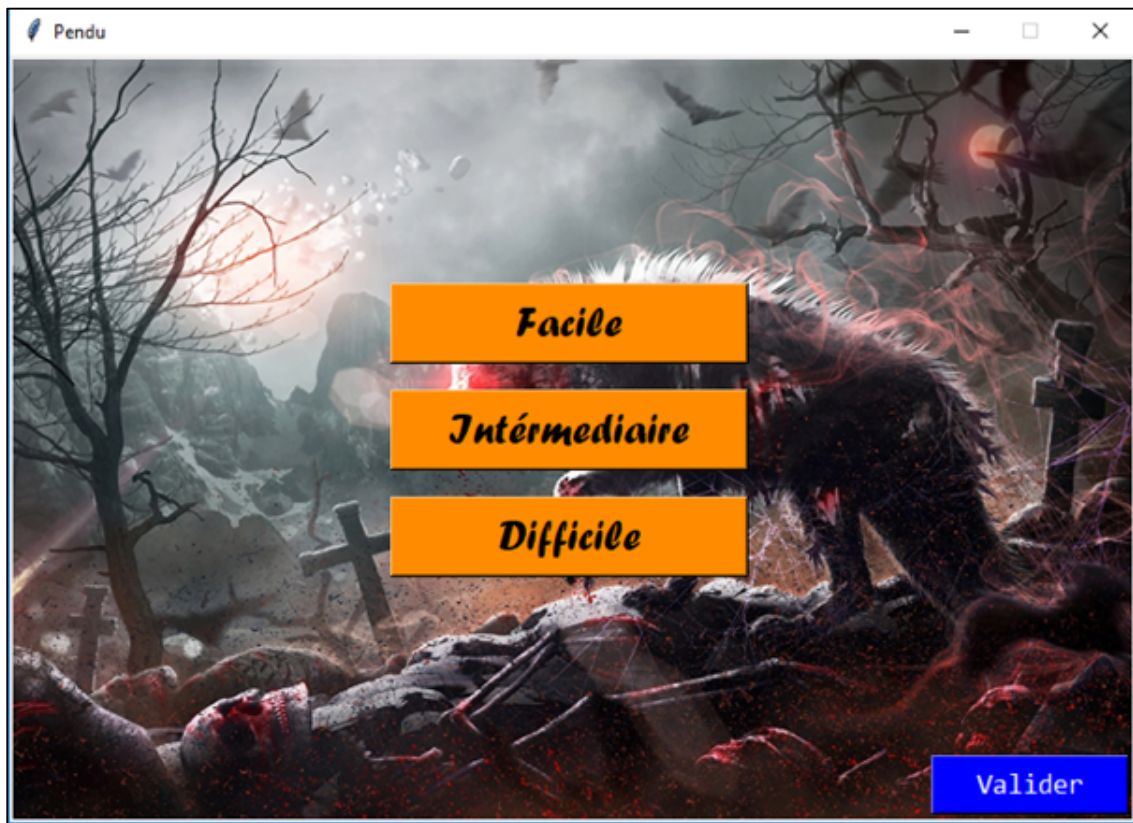
Esquisse de la première fenêtre du jeu :



(cf annexe pour le script/programme de l'esquisse)

Ce qu'elle est maintenant (2 fenêtres distinctes) :





V] BILAN ET PERSPECTIVES.

À niveau des améliorations possibles de notre, l'ajout de la musique de fond aurait été intéressant pour dynamiser notre jeu.

Nous avons également pensé à l'ajout d'une petite histoire introductive en tout début de partie pour instaurer une certaine atmosphère au jeu.

L'un des principaux défauts de ce programme se trouve aussi notamment dans la structure du code, en effet il est beaucoup trop long et le développeur qui l'analyse peut se perdre à l'intérieur, nous pensons qu'il faudrait à tout prix améliorer cet aspect du programme.

Chacun de nous, nous pensons que ce projet nous a apporté beaucoup de connaissance en programmation dans le langage Python, et nous avons été certainement surpris de ce qu'il pouvait faire, de jusqu'où il pouvait tout comme ses limites (par exemple, l'animation des avatars). En ce qui concerne les méthodes de travail, ce projet nous a permis d'être rigoureux dans la programmation afin de ne faire aucune erreur, de nous surpasser en tant qu'élève, d'être organisé dans la mesure où si nous avons quelque chose à terminer absolument pour la séance suivante, il fallait à tout prix respecter ce délai et surtout, de ne pas hésiter à faire appel aux autres si nous avons un quelconque problème, car c'est bien là tout l'intérêt d'un travail en groupe. Ce projet, et plus généralement la spécialité ISN nous a permis de consolider nos bases non seulement en programmation mais aussi en matière de communication au sein d'un groupe, d'entente et d'entraide mutuelle. Pour finir, nous tenions à remercier nos 2 professeurs pour nous avoir fournis l'aide nécessaire à la réalisation de ce projet, sans lesquels nous n'aurions pu en venir à bout, car nous sommes très satisfaits de notre rendu final.

VI] ANNEXES.

 Code de l'esquisse.

```
# -*- coding: utf-8 -*-
from tkinter import *
from random import*
import tkinter.font as tkFont
import tkinter as Tk

fenetre = Tk.Tk()
fenetre.title('Pendù')
fenetre.geometry("1012x600")
fenetre.resizable(width=False,height=False)

#-----LES FONCTIONS-----

def recup_nom():
    global Nom_utilisateur
    Nom_utilisateur=entree.get()
    lab06.config(text=str(Nom_utilisateur))

def fermer():
    recup_nom()
    # Si aucun prénom/pseudo n'est tapé :
    if Nom_utilisateur=="":
        value.set("ENTREZ VOTRE PRENOM D'ABORD !")
    elif Nom_utilisateur!="":
        fenetre.destroy()

def niv_fac():
    niveau=0
    niveau=1      # servira pour le bouton niveau facile
                  # (et pour la suite du programme, 1 est
associé          # au niveau facile).
    return niveau

def niv_int():
    niveau=0      # servira pour le bouton niveau intermédiaire
                  # (et pour la suite du programme, 2 est
associé          # au niveau intermédiaire).
    niveau=2
    return niveau

def niv_dif():
    niveau=0      # servira pour le bouton niveau difficile
                  # (et pour la suite du programme, 3 est
associé
```

```

                                # au niveau difficile).
    niveau=3
    return niveau

#-----L'INTERFACE -----

# POUR LA POLICE DE CARACTERE.
at20=tkFont.Font(fenetre, family='American Typewriter', size=20)
at15=tkFont.Font(fenetre, family='American Typewriter', size=15)
at15red=tkFont.Font(fenetre, family='American Typewriter', size=15)
at12=tkFont.Font(fenetre, family='American Typewriter', size=12)

# POUR LES PHOTOS A IMPORTER ET A OUVRIR (format gif).
photo = PhotoImage(file='niveau facile.gif')
photo2 = PhotoImage(file='niveau int.gif')
photo3 = PhotoImage(file='niveau dif.gif')
photo4 = PhotoImage(file='a.gif')
photo5 = PhotoImage(file='b.gif')
photo6 = PhotoImage(file='d1.gif')

# POUR LES LABELS.
lab01=Label(fenetre, image=photo4)
lab01.grid(row=1, column=4)
lab05=Label(fenetre, image=photo6)
lab05.grid(row=1, column=1, rowspan=40)

lab02=Label(fenetre, text="Entrez votre prénom :", font=at15)
value = StringVar()
entree = Entry(fenetre, width=100, textvariable=value,
justify='center', font=at12)
bou1 = Button(fenetre, text=' Valider.
', command=recup_nom, font=at12, pady=25)
bou1.grid(row=10, column=4)

lab02.grid(row=2, column=4, pady=20)
entree.grid(row=5, column=4)
lab03=Label(fenetre, font=at15, justify='center', text='Veuillez
choisir un niveau,')
lab03.grid(row=11, column=4)

lab06=Tk.Label(font=at15)
lab06.grid(row=12, column=4)

lab07=Tk.Label(font=at15red)

# POUR L'AFFICHAGE DES PHOTOS DES 3 NIVEAUX ET LES COMMANDES
CORRESPONDANTS.
niv_facile=Button(fenetre, image=photo, command=niv_fac)
niv_facile.grid(row=15, column=4)
niv_int=Button(fenetre, image=photo2, command=niv_int)

```

```


niv_int.grid(row=16, column=4)
niv_dif=Button(fenetre,image=photo3, command=niv_dif)
niv_dif.grid(row=17, column=4)

bou2 = Button(fenetre,font=at12,text="  Partir à l'aventure !  ",
command=fermer)
bou2.grid(row=18, column=4)

lab04=Label(fenetre, image=photo5)
lab04.grid(row=19, column=4, pady=50)

fenetre.mainloop()

```

 Code du jeu intégral :

```

# -*- coding: utf-8 -*-
from tkinter import *
import tkinter.font as tkFont
from timeit import default_timer
from random import*
import os.path
import pickle
from tkinter import messagebox

#_____CREATION DE VARIABLES DONT ON AURA
BES0IN._____

niveau=0
niveau=0
bouton=0
dico_avatar,dico_fond=[],[]
nom_utilisateur=""
masquer_fenetre=0
nombre_chance=10
mot_deviner=""
mot_trouve=""
score='0000'
liste_niveau=[]
bouton=[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25]
alphabet=['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z']
alphabet_mot=[]
d=[]
y=0
listeboutons=[]
niveau_facile=[]

```

```

niveau_intermediaire=[]
niveau_difficile=[]
compteur,compteur_final=0,0
classement_des_scores=[]
classement_des_noms=[]

# Le code est à lire de bas en haut.

#_____TROISIEME FENETRE_____

def recup_nom(): # fonction qui permet de récupérer le pseudo/nom de
l'utilisateur.
    global nom_utilisateur
    nom_utilisateur=demande_nom.get()
    if nom_utilisateur=="":
        nom_utilisateur="Joueur 1" # nom de l'utilisateur par défaut
si aucun pseudo n'est entré.

def classement():
    global pseudo_score,classement_des_noms,classement_des_scores
    classement_des_scores=[]
    classement_des_noms=[]
    if os.path.exists('donnees'): # Le fichier existe
        f = open('donnees',"rb") #On ouvre le fichier
        pseudo_score=pickle.load(f)# On recupère le dico dans
donnees contenant les pseudos avec les scores correspondants
        sored = sorted(pseudo_score.items(), key=lambda value:
value[1])# On trie le dico par ordre décroissant
        for pseudo,score in pseudo_score.items():
            classement_des_noms.append(pseudo)# On rajoute dans la
liste classement_des_noms les pseudos
            classement_des_scores.append(score)# On rajoute dans la
liste classement_des_score les scores

def recup_ancien_score():
    global score
    global pseudo_score
    global nom_utilisateur
    if os.path.exists('donnees'): # Le fichier existe
        # On le récupère
        f = open('donnees',"rb")
        pseudo_score=pickle.load(f)#On recupère le dico dans donnees
contenant les pseudos avec les scores correspondants
        if nom_utilisateur in pseudo_score.keys():
            score=int(pseudo_score[nom_utilisateur])#On récupère le
score correspondant au nom d'utilisateur si son nom est déjà dans le
fichier donnees
        else:
            score='0000'#Sinon on lui créé un score initialiser
à 0

```



```

        pseudo_score[nom_utilisateur]=score#on lui créer un
pseudo et un score dans le dictionnaire

    f.close()
else: # Le fichier n'existe pas
    score='0000'#0n initialise le score à 0
    pseudo_score={}#0n créer un dico pseudo_score
    pseudo_score[nom_utilisateur]=score#0n rajoute dans le dico
le nom d'utilisateur avec le score à 0 correspondant

def enreg_score():
    global pseudo_score
    global nom_utilisateur
    global score
    if os.path.exists('donnees'): # Le fichier existe
        # 0n le récupère
        f = open('donnees',"rb")
        pseudo_score=pickle.load(f)#0n recupère le dico dans donnees
contenant les pseudos avec les scores correspondants
        f.close()
        pseudo_score[nom_utilisateur]=int(score)#0n récupère le
score en type entier
        f = open('donnees', "wb") # 0n écrase les anciens scores
        pickle.dump(pseudo_score,f)
        f.close()

    else:
        pseudo_score={}#le fichier n existe pas, on créer un dico
pseudo_score
        pseudo_score[nom_utilisateur]=int(score)#0n créé un nom
d'utilisateur au joueur dans le dico avec le score qu'il a réalisé
        f = open('donnees', "wb") # 0n écrase les anciens scores
        pickle.dump(pseudo_score,f)#0n dépose le dico dans le
fichier donnees
        f.close()

# Fonctions gérant les éléments saisis par l'utilisateur
def avatar():
    global dico_avatar,dico_fond
    Avatar_1=PhotoImage(file="Image/1.gif")
    Avatar_2=PhotoImage(file="Image/2.gif")
    Avatar_3=PhotoImage(file="Image/3.gif")
    Avatar_4=PhotoImage(file="Image/4.gif")
    Avatar_5=PhotoImage(file="Image/5.gif")
    fond1=PhotoImage(file="Image/1499730799-pendue.gif")
    fond2=PhotoImage(file="Image/Cemetery_Wolves_Gothic_Fantasy_by_M
r-Ripley_Cross_527820_1280x719.gif")
    img=PhotoImage(file="Image/circle-1544536_960_720.gif")
    img2=PhotoImage(file="Image/Ligne-de-séparation.gif")

```

```

    img3=PhotoImage(file="Image/square-frame-
11526207112sccmpzpuzkp.gif")
    dico_avatar=[Avatar_1,Avatar_2,Avatar_3,Avatar_4,Avatar_5]
    dico_fond=[fond1,fond2,img,img2,img3]

def valider():
    recup_nom()
    #score_fct()
    creation_fenetre2(dico_fond[1])
    fenetre.withdraw() #permet de cacher la première fenetre (choix
du pseudo, avatar...)
                        #pour afficher la deuxième fenetre

def bou1(): # affectation d'un certain nombre en fonction de
l'avatar choisi par l'utilisateur.
    global bouton
    bouton=1
    choix_bouton()

def bou2():
    global bouton
    bouton=2
    choix_bouton()

def bou3():
    global bouton
    bouton=3
    choix_bouton()

def bou4():
    global bouton
    bouton=4
    choix_bouton()

def bou5():
    global bouton
    bouton=5
    choix_bouton()

def choix_bouton(): #permet de changer la couleur du bouton
    for i in range(len(liste_bouton)):
        liste_bouton[i].configure(bg="WHITE") #tous les boutons
sont en blancs au début
        liste_bouton[bouton-1].configure(bg="RED") #le bouton cliqué
devient rouge. On commence à -1 car la liste
                                                    # commence à 0.

#_____DEUXIEME
FENETRE_____

```

```

def creation_fenetre2(fond2):

    def niv_fac(): # creation des fonctions pour les niveaux,
affectation d'un certain nombre en fonction
        # du niveau choisi par l'utilisateur.
        global niveau
        niveau=1
        choix_niveau(liste_niveau)
    def niv_int():
        global niveau
        niveau=2
        choix_niveau(liste_niveau)
    def niv_dif():
        global niveau
        niveau=3
        choix_niveau(liste_niveau)
    def choix_niveau(liste):
        for i in range(len(liste)):
            liste[i].configure(bg="ORANGE") # tous les boutons sont
en orange au début
            liste[niveau-1].configure(bg="RED") # le bouton cliqué
devient rouge. On commence à -1 car la liste
                                                # commence à 0.

    def valider2():
        global masquer_fenetre
        if niveau==0:
            messagebox.showerror("Erreur", "Veuillez sélectionner un
niveau")
        else:
            creation_fenetre3(dico_fond[2],dico_fond[3]) # images de
fonds utilisés en paramètre
            masquer_fenetre=1 # affecte la valeur 1 à
masquer_fenetre (au début = 0). Masquer_fenetre prend la valeur 0
            # si aucun niveau n'a été sélectionné.
Si la valeur 0 est resté, c'est que l'utilisateur
            # n'a pas cliqué sur un bouton niveau.
            fenetre2.withdraw() #.withdraw() permet de cacher la
deuxième fenetre (choix niveau).

    #-----Caractéristiques de la deuxième fenetre :

    fenetre2=Toplevel() # permet de se focaliser, porter l'attention
sur la deuxième fenetre
    fenetre2.grab_set()
    fenetre2.focus_set()
    fond_c2=Canvas(fenetre2,width=735,height=500)
    fond_c2.create_image(295,253,image=fond2) #image de fond de la
deuxième fenetre.
    fenetre2.resizable(width=False,height=False) # ne permet pas de
redimensionner la fenetre.

```

```

    # création des boutons niveau facile, intermédiaire et
difficile.
    niv_facile_boutton=Button(fenetre2,text="Facile",font="Forte
20", command=niv_fac,bg="DARK ORANGE",width=15)
    niv_intemerdiaire_boutton=Button(fenetre2,
text="Intermédiaire",font="Forte 20", command=niv_int,bg="DARK
ORANGE",width=15)
    niv_difficile_boutton=Button(fenetre2,text="Difficile",font="Fort
e 20",command=niv_dif,bg="DARK ORANGE",width=15)

    # on rajoute dans les listes qui correspondent à chaque niveaux
les mots de chaque niveau.
    liste_niveau.append(niv_facile_boutton),liste_niveau.append(niv_
intemerdiaire_boutton),liste_niveau.append(niv_difficile_boutton)
    bt_valider2=Button(fenetre2,text="Valider",font="Consolas
14",command=valider2,fg="WHITE",bg="BLUE",width=12)
    niv_facile_boutton.place(x=250,y=150)
    niv_intemerdiaire_boutton.place(x=250,y=220)
    niv_difficile_boutton.place(x=250,y=290)
    bt_valider2.place(x=605,y=460)
    fond_c2.pack()

# _____ CREATION DE LA TROISIEME
FENETRE _____

def creation_fenetre3(img,img2):

    def creation_dico_mot():
        f=open("1.txt",'r') #On va lire le fichier .txt
        c="" #On créer une variable c de type str qui contiendra
tous les caractères du fichier.txt
        for ligne in f: #On parcourt toutes les lignes dans f
            c=c+ligne #On met dans la variable c toutes les lignes
de f
        f.close()
        a=c.split()
        i=0
        while i!=len(a): # en fonction de la longueur du mot,
attribuer un niveau spécifique :
            if len(a[i])<=6: # si le mot fait au plus 6 lettres,
c'est le niveau facile
                niveau_facile.append(a[i]) # l'ajouter dans la liste
niveau_facile qui nous servira plus tard.
                i=i+1 # incrémentation.
            elif 6<len(a[i])<=10: # si le mot fait entre 6 et 10
lettres, c'est le niveau intermédiaire.
                niveau_intermediaire.append(a[i]) # l'ajouter dans
la liste niveau_intermediaire qui nous servira plus tard.
                i=i+1 # incrémentation.

```

```

        else:    ## si le mot fait plus de 10 lettres, c'est le
niveau difficile.
            niveau_difficile.append(a[i]) # l'ajouter dans la
liste niveau_difficile qui nous servira plus tard.
            i=i+1

    def choix_mot():    # choix du mot que l'utilisateur devinera en
fonction du niveau cliqué
        global mot_deviner,mot_trouve
        if niveau==1:    # comme le bouton niveau facile est associé
au nombre 1, on choisira un mot au hasard dans la liste
niveau_facile
            mot_deviner=choice(niveau_facile)
            mot_deviner=mot_deviner.upper()
        elif niveau==2: # comme le bouton niveau intermédiaire est
associé au nombre 2, on choisira un mot au hasard dans la liste
niveau_intermédiaire
            mot_deviner=choice(niveau_intermediaire)
            mot_deviner=mot_deviner.upper()
        elif niveau==3: # comme le bouton niveau difficile est
associé au nombre 3, on choisira un mot au hasard dans la liste
niveau_difficile.
            mot_deviner=choice(niveau_difficile)
            mot_deviner=mot_deviner.upper()

    def updateTime(): # création du chronomètre.
        now = default_timer() - start # prend le temps par défaut, à
ce moment là, que l'on soustrait à start qui est aussi la valeur par
défaut
                                                # permet donc d'initialiser la
variable à 0h 00 min 00 s.
        minutes, seconds = divmod(now, 60) # permet de compter en
base de 60
        hours, minutes = divmod(minutes, 60)
        str_time = "%d:%02d:%02d" % (hours, minutes, seconds)
#séparation en heure, minute et secondes.
        fond_c3.itemconfigure(text_clock, text=str_time,font
=my_font3) # permet de créer l'horloge : 00:00:00
        fenetre3.after(1000, updateTime) #création d'un widget
dynamique qui permet d'initialiser le widget à chaque seconde, màj à
chaque seconde.

    def trace_pendu(nombre_chance): # création du dessin du pendu en
fonction du nombre de chances restants à l'utilisateur
        if nombre_chance<=9:
            dessin_pendu.create_line(50,390,250,390,width=4)
        if nombre_chance<=8:
            dessin_pendu.create_line(150,390,150,100,width=4)
        if nombre_chance<=7:
            dessin_pendu.create_line(150,100,300,100,width=4)
            dessin_pendu.create_line(150,145,200,100,width=4)

```



```

    if nombre_chance<=6:
        dessin_pendu.create_line(300,100,300,150,width=4)
    if nombre_chance<=5:
        dessin_pendu.create_oval(270,150,330,200,width=4)
    if nombre_chance<=4:
        dessin_pendu.create_line(300,200,300,290,width=4)
    if nombre_chance<=3:
        dessin_pendu.create_line(300,210,270,250,width=4)
    if nombre_chance<=2:
        dessin_pendu.create_line(300,210,330,250,width=4)
    if nombre_chance<=1:
        dessin_pendu.create_line(300,290,265,330,width=4)
    if nombre_chance<=0:
        dessin_pendu.create_line(300,290,335,330,width=4)

    def mot_afficher(): # permet de savoir à quel moment
l'utilisateur a réussi à deviner le mot ou pas
        global mot_trouve
        mot_trouve=""
        for lettre in mot_deviner: # pour chaque lettre dans le mot
qui a été tiré au sort
            if lettre in alphabet_mot: # si la lettre dans
l'alphabet
                mot_trouve=mot_trouve+lettre
            else:
                mot_trouve=mot_trouve+"-"

    def affichage_mot_deviner(): # permet d'afficher la lettre à la
place du tiret correspondant
        global d
        mot_deviner_frame=LabelFrame(fenetre3, width=1080,
height=40,bd=0)
        r=0
        for i in mot_deviner:
            d.append(r)
            r=r+1

        t=0
        for i in range (len(mot_deviner)):
            d[i]=Label(mot_deviner_frame, text='-
',width=5,bg="BEIGE",font="Arial 20")
            d[i].place(x=540-(len(mot_deviner)*30)+t,y=7)
            t=t+60 # permet de placer les labels pour éviter les
superpositions

        mot_deviner_frame.place(x=0,y=160)

    def creation_boutton(cadre): # permet de créer les 26 boutons de
l'alphabet
        for i in range(0,26):
            if i%6==0:
                y=0

```

```

        if i<6: # création des 6 première lettres
            bouton[i]=Button(cadre,
text=alphabet[i],bd=5,width=8,padx=12,pady=12,command=lambda x=i:
change(x),disabledforeground="BLACK")
                                #disabledforeground
permet de désactiver le bouton une fois cliqué, sa couleur reste
noire

            bouton[i].grid(row=0,column=y)
        if 6<=i<12: # création des 6 prochaines ect...
            bouton[i]=Button(cadre,
text=alphabet[i],bd=5,width=8,padx=12,pady=12,command=lambda x=i:
change(x),disabledforeground="BLACK")
            bouton[i].grid(row=1,column=y)
        if 12<=i<18:
            bouton[i]=Button(cadre,
text=alphabet[i],bd=5,width=8,padx=12,pady=12,command=lambda x=i:
change(x),disabledforeground="BLACK")
            bouton[i].grid(row=2,column=y)
        if 18<=i<24:
            bouton[i]=Button(cadre,
text=alphabet[i],bd=5,width=8,padx=12,pady=12,command=lambda x=i:
change(x),disabledforeground="BLACK")
            bouton[i].grid(row=3,column=y)
        if 24<=i<=25: # il ne reste plus que y et z sur la
cinquième ligne
            bouton[i]=Button(cadre,
text=alphabet[i],bd=5,width=8,padx=12,pady=12,command=lambda x=i:
change(x),disabledforeground="BLACK")
            bouton[i].grid(row=4,column=y+2)

        listeboutons.append(bouton[i])
        y=y+1

def change(i):
    global nombre_chance,compteur_final
    if mot_deviner!=mot_trouve and nombre_chance>0: # si le mot
deviné est différent du mot trouvé et si le nombre
                                                    # de chances
supérieur à 0 (l'utilisateur n'a pas encore joué)
        if alphabet[i] in mot_deviner:
            listeboutons[i].config(bg='light green') # pour
chaque lettre de l'alphabet qui se situe dans le mot à deviner
                                                    # le bouton
devient vert
            bouton[i].configure(state='disabled') # une fois
cliqué, le bouton est désativé, pour limiter les bugs
            alphabet_mot.append(alphabet[i]) # rajout
dans une liste que l'on utilisera pour savoir si l'utilisateur
                                                    # a déjà
trouvé le même mot.

```

```

        mot_afficher() # appel à
une fonction qui permet d'afficher le mot
    else: # si la
lettre n'est pas dans le mot à deviner
        listeboutons[i].config(bg='red') # le bouton
devient rouge
        bouton[i].configure(state='disabled') # une fois
cliqué, le bouton est désactivé, pour limiter les bugs
        nombre_chance=nombre_chance-1
        trace_pendu(nombre_chance) # traçage
du pendu

    n=0 # initialisation des rangs à 0
    while n !=len(mot_deviner): # tant que n est différent de la
longueur du mot à deviner :
        if alphabet[i]==mot_deviner[n]: # si la lettre de
l'aphabet est dans le mot à deviner
            d[n].config(text=alphabet[i]) # on change le tiret
correspondant par la lettre

            n=n+1 # incrémentation
        else: # si la lettre n'est pas dans le mot à deviner
            n=n+1 # on incrémente tout simplement.

    if nombre_chance==0: # si le nombre de chance = 0,
l'utilisateur perd
        compteur_final=compteur # permet d'arrêter le chrono,
sauvegarde du temps à l'instant même
        dessin_pendu.configure(bg="RED") # le canva devient
rouge
        fenetre_defaite_fct(dico_fond[2],dico_avatar[boutton-1])
# images de fond en paramètre
    elif mot_deviner==mot_trouve: # si l'utilisateur a gagné :
        compteur_final=compteur # permet d'arrêter le chrono,
sauvegarde du temps à l'instant même
        dessin_pendu.configure(bg="GREEN") # le canva devient
vert
        attribution_score(niveau) # donner un score à la
personne
        fenetre_victoire(dico_fond[2],dico_avatar[boutton-1])
#fenetre victoire s'affiche avec utilisation de fonds

#-----CREATION D'UN SYSTEME DE CLASSEMENT DANS LA TROISIEME
FENETRE

    def affichage_classement(fenetre): # permet de créer 3 listes
dans lesquelles on va rajouter autant d'élément
        # qu'il y en a dans la liste
classement_des_noms

```

```

        global
classement_des_noms,pseudo_score,classement_des_scores
        c=[]
        e=[]
        f=[]
        x=1
        y=1
        classement_frame=LabelFrame(fenetre,width=1080,height=4,bd=1
0)
        r=0
        for i in classement_des_noms:
            c.append(r)
            e.append(r)
            f.append(r)
            r=r+1 # on incrémente r pour avoir des éléments
différents
        for i in range(len(classement_des_noms)): # Pour chaque
valeur dans classement_des_noms
            c[i]=Label(classement_frame,
text=classement_des_noms[i], width=10, bg='BEIGE', font='Arial 20')
            # On prend chaque élément de la liste c et on
créer pour chacun un label qui affichera le nom du joueur.
            c[i].grid(row=x,column=1)
            e[i]=Label(classement_frame,
text=classement_des_scores[i], width=10, bg='BEIGE', font='Arial
20')
            # On prend chaque élément de la liste e et on crée
pour chacun un label qui affichera le score du joueur.
            e[i].grid(row=x,column=2)
            e[i]=Label(classement_frame, text=x, width=10,
bg='BEIGE', font='Arial 20')
            # On prend chaque élément de la liste e et on crée
pour chacun un label qui affichera le rang du joueur.
            e[i].grid(row=x,column=0)
            x=x+1 # on incrémente x, pour qu'il ne reste pas à 1.

        # placements des labels qui correspondent chacun au texte
"Rang", "Joueur" et "Score".
        rang=Label(classement_frame, text='Rang', width=10,
bg='BEIGE', font='Arial 20',bd=10)
        rang.grid(row=0,column=0)
        Joueur=Label(classement_frame, text='Joueur', width=10,
bg='BEIGE', font='Arial 20',bd=10)
        Joueur.grid(row=0,column=1)
        Score=Label(classement_frame, text='Score', width=10,
bg='BEIGE', font='Arial 20',bd=10)
        Score.grid(row=0,column=2)
        classement_frame.place(x=120,y=250) # placement du
classement des 3 premiers

def affichage_3_premiers():

```

```

        global
classement_des_noms,pseudo_score,classement_des_scores # même
principe ici.
        c=[]
        e=[]
        f=[]
        x=1
        y=1
        classement_frame=LabelFrame(fenetre3,width=1080,height=4,bd=
10,bg="BEIGE")
        r=0
        for i in classement_des_noms:
            c.append(r)
            e.append(r)
            f.append(r)
            r=r+1
        if len(classement_des_noms)>=3: # s'il y a plus de 3
éléments dans classement_des_noms
            for i in range(0,3): # On n'a affiche que les 3 premiers
éléments .
                c[i]=Label(classement_frame,
text=classement_des_noms[i], width=10, bg='BEIGE', font='Arial 10')
                c[i].grid(row=x,column=1)
                e[i]=Label(classement_frame,
text=classement_des_scores[i], width=10, bg='BEIGE', font='Arial
10')
                e[i].grid(row=x,column=2)
                e[i]=Label(classement_frame, text=x, width=10,
bg='BEIGE', font='Arial 10')
                e[i].grid(row=x,column=0)
                x=x+1 # on incrémente le rang de sorte que ça ne
reste pas à 1.

                # placements des labels qui correspondent chacun au
texte "Rang", "Joueur" et "Score".
                rang=Label(classement_frame, text='Rang', width=10,
bg='BEIGE', font='Arial 10',bd=10)
                rang.grid(row=0,column=0)
                Joueur=Label(classement_frame, text='Joueur',
width=10, bg='BEIGE', font='Arial 10',bd=10)
                Joueur.grid(row=0,column=1)
                Score=Label(classement_frame, text='Score',
width=10, bg='BEIGE', font='Arial 10',bd=10)
                Score.grid(row=0,column=2)
                classement_frame.place(x=420,y=10) # placemment du
classement des 3 premiers

#-----CREATION D'UNE FONCTION PRINCIPALE

```



```

def fonction_principale(start): # pour ne pas réécrire les 8
fonctions.
    creation_dico_mot()
    choix_mot()
    affichage_mot_deviner()
    creation_boutton(cadre_boutton)
    updateTime()
    incremente()
    classement()
    affichage_3_premiers()

def incremente(): #création d'une variable compteur qui
s'incréméte en même temps que le chrono
    # pour récupérer le temps.
    "Incréméte le compteur à chaque seconde"
    global compteur
    compteur += 1
    fenetre3.after(1000, incremente)

def attribution_score(niveau): # permet de donner un score à
l'utilisateur
    global score
    score=int(score)
    recup_ancien_score()
    score=int(score)
    if compteur_final<30: # le score dépend du temps qu'a mis
l'utilisateur à trouver ou non le mot à deviner
        score=niveau*1000+score # en moins de 30 secondes, 1000
points
    elif compteur_final<60 and compteur_final>30: # entre 30 et
60 secondes, 500 points
        score=niveau*500+score
    else:
        score=niveau*100+score # un temps plus grand, 100
points.
    enreg_score()

    fenetre3=Toplevel() # permet de se focaliser sur la troisième
fenetre.
    fenetre3.grab_set()
    fenetre3.focus_set()
    fenetre3.geometry("1080x600")
    fenetre3.resizable(width=False,height=False)
    my_font2=tkFont.Font(fenetre3,family="Cataneo BT",size=25)
    my_font3=tkFont.Font(fenetre3,family="Cataneo BT",size=35)
    fond_c3=Canvas(fenetre3,width=1080,height=600)
    cadre_boutton=LabelFrame(fenetre3, width=600, height=600,bd=0)

    recup_ancien_score() # on récupère les anciens scores que l'on
affichera ensuite dans un
    # petit tableau en haut de la fenêtre.

```

```

    affichage_score=Label(fenetre3,text=
pseudo_score[nom_utilisateur],font=my_font2) #
    affichage_nom=Label(fenetre3,text=nom_utilisateur,font=my_font2)
    fond_c3.create_image(80,80,image=dico_avatar[boutton-1])
    fond_c3.create_image(80,80, image=img)
    fond_c3.create_image(540,160,image=img2)
    fond_c3.create_text(200,115,text="Score:",font=my_font2)
    text_clock = fond_c3.create_text(900,90)
    dessin_pendu=Canvas(fenetre3,width="400",height="400",bd=10)
    dessin_pendu.create_image(200,225,image=dico_fond[4])
    start = default_timer()

    fonction_principale(start) # appel à la fonction start qui
activera elle-même 8 autres fonctions.

    fond_c3.place(x=10,y=20)
    affichage_score.place(x=270,y=118)
    affichage_nom.place(x=190,y=40)
    dessin_pendu.place(x=20,y=180)
    cadre_boutton.place(x=500,y=280)

#
def reactiver(): # réactivation de tous les boutons, car ils
étaient désactivés (disabledforeground)
    # pour limiter les bugs
    for k in range(26):
        bouton[k].configure(state='normal')

def reinitialiser(): # réinitiation si le joueur veut rejouer :
réinitialisation de toutes les variables modifiées auparavant
    global
niveau,masquer_fenetre,nombre_chance,mot_deviner,mot_trouve,liste_ni
veau,alphabet_mot,d,y,listeboutons,compteur_final, compteur
    niveau=0 # niveau redevient 0
    masquer_fenetre=0 # aucune fenetre masquée
    nombre_chance=10 # nombre de chance réinitialisé à 10
    mot_deviner=""
    mot_trouve=""
    liste_niveau=[]
    alphabet_mot=[]
    d=[]
    y=0
    listeboutons=[]
    compteur=0 # temps remis à 0
    compteur_final=0 # temps remis à 0
    reactiver() # fonction pour réactiver tous les boutons

```

```

def fenetre_defaite_fct(img,img2): # si le joueur veut
réessayer/rejouer
    def rejouer():
        reinitialiser() # appel de la fonction reinitialiser
        fenetre3.destroy() #destruction de la fenetre 3
        fenetre_defaite.destroy()
        creation_fenetre2(dico_fond[1]) # on retourne à la
deuxième fenêtre pour refaire un choix de niveau

        fenetre_defaite=Toplevel() # fenetre pour informer que
l'utilisateur a perdu
        fenetre_defaite.grab_set()
        fenetre_defaite.focus_set()
        fenetre_defaite.geometry("900x600") # dimensions de la
fenetre_defaite
        fenetre_defaite.resizable(width=False,height=False)
        my_font2=tkFont.Font(fenetre_defaite,family="Cataneo
BT",size=25)
        fond_c4=Canvas(fenetre_defaite,width=800,height=200)
        fond_c4.create_text(420,20,text="Dommage "+nom_utilisateur+"
votre mot était: "+mot_deviner,font=my_font2)
        enreg_score()
        fond_c4.create_text(425,60,text="vous gagnerez une prochaine
fois",font=my_font2)
        fond_c4.create_text(150,120,text="Score: ",font=my_font2)
        rejouer_bt=Button(fenetre_defaite,text="Rejouer",command=rej
ouer,width=10,font="Forte 20",bg="GREEN")
        quitter=Button(fenetre_defaite,text="Quitter",command=fenetr
e.destroy,width=10,font="Forte 20",bg="RED")
        affichage_classement(fenetre_defaite) # appel à la fonction
pour pouvoir regarder le classement dans la fenetre défaite.
        affichage_score=Label(fenetre_defaite,text=score,font=my_fon
t2)

        rejouer_bt.place(x=420,y=530)
        quitter.place(x=600,y=530)
        affichage_score.place(x=260,y=95)
        fond_c4.pack()

def fenetre_victoire(img,img2): # si le joueur veut réessayer,
même fonctionnement
    def rejouer2():
        reinitialiser() # appel de la fonction reinitialiser
        #score_fct()
        recup_ancien_score() # permet de récupérer le score que
le joueur a eu, pour ensuite pouvoir rejouer avec le même score.
        creation_fenetre2(dico_fond[1]) # retour à la deuxième
fenêtre pour que le joueur choisisse un niveau
        fenetre3.destroy() # on détruit évidemment la troisième
fenetre, puis la fenetre_victoire.
        fenetre_victoire.destroy()

```

```

        fenetre_victoire=Toplevel() # permet de se focaliser sur la
fenetre victoire et de laisser tomber les autres fenetre.
        fenetre_victoire.grab_set()
        fenetre_victoire.focus_set()
        fenetre_victoire.geometry("900x600") # dimensions de la
fenetre_victoire.
        fenetre_victoire.resizable(width=False,height=False) # non
redimensionnable.
        my_font2=tkFont.Font(fenetre_victoire,family="Cataneo
BT",size=25)
        enreg_score()
        classement()
        fond_c4=Canvas(fenetre_victoire,width=500,height=200)
        fond_c4.create_text(250,20,text="Félicitation
"+nom_utilisateur,font=my_font2)
        fond_c4.create_text(250,50,text="vous avez
gagné",font=my_font2)
        fond_c4.create_text(200,120,text="Score: ",font=my_font2)
        affichage_classement(fenetre_victoire) # appel à la fonction
pour pouvoir regarder le classement dans la fenetre victoire.
        rejouer_bt=Button(fenetre_victoire,text="Rejouer",command=re
jouer2,width=10,font="Forte 20",bg="GREEN")
        quitter=Button(fenetre_victoire,text="Quitter",command=fenet
re.destroy,width=10,font="Forte 20",bg="RED")
        affichage_score=Label(fenetre_victoire,text=score,font=my_fo
nt2)

        rejouer_bt.place(x=420,y=530)
        quitter.place(x=600,y=530)
        affichage_score.place(x=500,y=95)
        fond_c4.pack()

# _____PREMIERE FENETRE (pseudo,
avatar...)._____
fenetre = Tk()
fenetre.title('Pendule')
fenetre.geometry("735x500")
fenetre.resizable(width=False,height=False) # fenetre impossible à
redimensionner.
my_font =tkFont.Font(fenetre, family="Vivaldi",size=60, underline =
1 )

fond_c=Canvas(fenetre,width=1012,height=600) # Création d'un Canvas
pour pouvoir ensuite écrire dessus.
fond = PhotoImage(file="Image/1499730799-pendue.gif")
fond_c.create_image(200, 300, image=fond) #image de fond
fond_c.create_text(367,50,text="Le pendu"
,font=my_font,fill="WHITE")

```

```

fond_c.create_text(550,140,text="Sous quel nom souhaitez-vous
jouer?",font="Consolas 14",fill="BEIGE")
demande_nom=Entry(fond_c,bg="BEIGE",fg="DARK RED",font='Helvetica 18
bold',width=17) # création de l'entrée pour mettre le pseudo de
l'utilisateur.
fond_c.create_text(490,230,text="Choisissez un
avatar:",font="Consolas 14",fill="BEIGE")

avatar()

bt_1 =
Button(fenetre,image=dico_avatar[0],activebackground="GREY",command=
bou1) # creation des boutons correspondent aux 5 avatars.
bt_2 =
Button(fenetre,image=dico_avatar[1],activebackground="GREY",command=
bou2)
bt_3 =
Button(fenetre,image=dico_avatar[2],activebackground="GREY",command=
bou3)
bt_4 =
Button(fenetre,image=dico_avatar[3],activebackground="GREY",command=
bou4)
bt_5 =
Button(fenetre,image=dico_avatar[4],activebackground="GREY",command=
bou5)

liste_boutton=[bt_1,bt_2,bt_3,bt_4,bt_5]
bt_valider=Button(fenetre,text="Valider",fg="WHITE",font="Consolas
14",bg="BLUE",command=valider,width=12)
bt_1.place(x=380,y=250) # placement des boutons
bt_2.place(x=490,y=250)
bt_3.place(x=600,y=250)
bt_4.place(x=435,y=345)
bt_5.place(x=560,y=345)
bt_valider.place(x=600,y=458)
fond_c.pack()
demande_nom.place(x=430,y=170)

fenetre.mainloop()

```