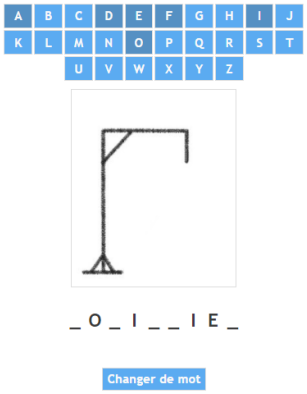
Le pendu

Tentez de deviner le mot secret en entrant des lettres une par une au clavier. Ne gaspillez pas vos coups, car si trop de vos choix sont erronés vous tuerez le pendu et vous perdrez la partie.
Site http://jeudupendu.fr/



La première chose à faire pour l'ordinateur est de disposer d'un mot pris au hasard dans une liste

La liste des mots est fournie : words.json

1. Version console

Le projet est réalisé en groupe de 2. Vous devrez vous répartir le travail avec une partie commune et des parties distinctes.

```
NOTEBOOK
Il vous reste  8
 vies et vous avez utilisé les lettres :
le mot est : - - - - - - - - -
ENTRER UN CARACTERE :
```

2. Avec Graphic Interface User

Source Wikipédia
Le motif MVC a été créé par Trygve Reenskaug lors de sa visite du Palo Alto Research Center (abr. PARC) en 19781. Le nom original est « thing model view » editor pattern, puis il a été rapidement renommé model-view-controller pattern

Le patron MVC a été utilisé la première fois pour créer des interfaces graphiques avec le langage de programmation Smalltalk en 1980.
Une application conforme au motif MVC comporte trois types de modules :

- Les modèles,
- Les vues
- Les contrôleurs2.

2.1. Modèle

Élément qui contient les données ainsi que de la logique en rapport avec les données : validation, lecture et enregistrement ; il peut, dans sa forme la plus simple, contenir uniquement une simple valeur, ou une structure de données plus complexe
Le modèle représente l'univers dans lequel s'inscrit l'application : Par exemple pour une application de banque, le modèle représente des comptes, des clients, ainsi que les opérations telles que dépôt et retraits, et vérifie que les retraits ne dépassent pas la limite de crédit.

Luc.vincent@ac-bordeaux.fr

2.2. Vue

Partie visible d'une interface graphique, la vue se sert du modèle, et peut être un diagramme, un formulaire, des boutons, etc. Une vue contient des éléments visuels ainsi que la logique nécessaire pour afficher les données provenant du modèle. Dans une application de bureau classique, la vue obtient les données nécessaires à la présentation du modèle en posant des questions. Elle peut également mettre à jour le modèle en envoyant des messages appropriés.

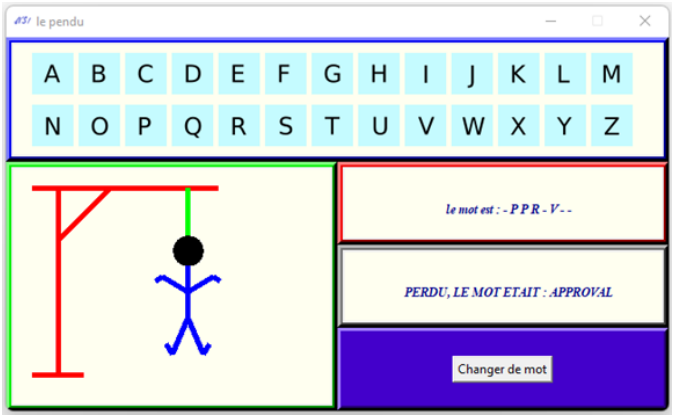
2.3. Contrôleur

Module qui traite les actions de l'utilisateur, modifie les données du modèle et de la vue

3. Travail envisagé

La réalisation d'une interface graphique est complexe et dépasse largement les objectifs du programme de première.

Le projet est fourni et vous ne devrez pas le modifier. Il vous restera à développer certains modules qui doivent fonctionner de façon autonome puis être inséré dans le projet complet.



a

root.py

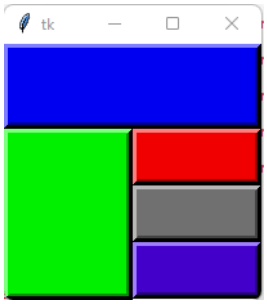
```
def create_main():
    '''
    Créer un objet fenêtre graphique des dimensions souhaitées
    Renvoyer un objet fenêtre graphique <class 'tkinter.Tk'>
    '''
```

Contraintes :

- La fenêtre n'est pas redimensionnable
- La fenêtre doit être créée en fonction de la taille de l'écran.
- Le nom et une icône doivent apparaître

frames.py

```
def create_frame(main_gui):
    '''
    Creer les zones d'affichages
    main_gui : <class 'tkinter.Tk'>
    Renvoie la liste des Frames [<tkinter.Frame object .!frame>, ]
    '''
```



Contraintes :

- Les frames (cadres) sont répartis selon l'organisation et la taille imposée
- Les couleurs permettent de les distinguer.
- La liste est ordonnée bleu, vert, rouge, gris, violet

vocabulaire.py

```
def open_json(cible):
    ''' Ouvrir un fichier json
    cible:str le nom du fichier à ouvrir
    rep: list[str] une liste de mots
    '''
```

Contraintes :

- Une liste des 2466 mots du fichier fourni est renvoyée

mot.py

```
def valid_word(words):
    '''
    renvoie un mot "valide" pris au hasard dans la liste words
    words : list[str]
    a_word : str un mot valide en majuscule
    '''
```

Contraintes :

- Un mot valide ne contient ni espace ni tiret. Par exemple 'ill-fated' ou 'mess up' ne peuvent pas être retenu pour jouer.

display.py

```
def add_text(frame, message):
    ''' Renvoie le message dans un canvas placé dans le frame'''
```

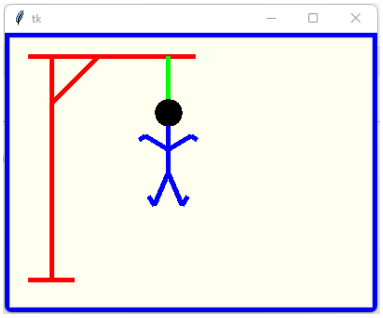
Contraintes :

- Renvoie un canvas (cadre) contenant le message
- Le canvas est un objet du frame passé en argument
- Le canvas a une couleur imposée pour le repérer
- Le message st affiché au centre du canvas

Luc.vincent@ac-bordeaux.fr

draw.py

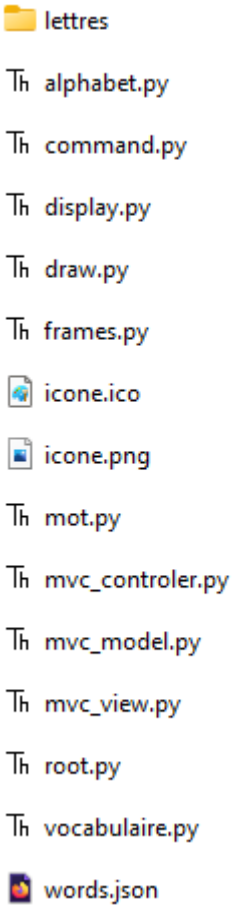
```
def add_pendu(frame, level):
    '''
    Dessiner dans un canvas du frame
    le pendu au niveau level (x/15) au max
    '''
```



Contraintes :

- Renvoie un canvas (cadre) contenant le dessin
- Le dessin est partiellement dessiné selon le niveau variant de rien à tout (0 à 15)
- Il peut être utile de vider le frame avant de créer le nouveau canvas.
- Le dessin doit pouvoir s'adapter à la taille disponible

4. *Assemblage final*



Le programme mvc_controleur.py doit vous permettre de jouer.