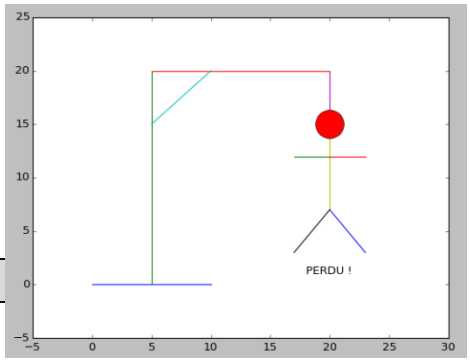
DM 2 - PROJET INFORMATIQUE

LE JEU DU PENDU



*Tout le travail demandé dans ce paragraphe doit être contenu dans un fichier nommé **DM02_Pendu_VotreNom.py** que vous déposerez sur le site prepabellevue.org rubrique PCSI2/Informatique/DM*

Le but de ce deuxième projet est de programmer un jeu du pendu.

On utilisera un lexique (à télécharger sur prepabellevue.org, rubrique PCSI2/Informatique/DM) pour avoir suffisamment de mots pour jouer.

1. AFFICHAGE

Commencez par copier/coller la liste des segments à tracer

```
L=[[[0,10],[0,0]],[[5,5],[0,20]],[[5,20],[20,20]],[[5,10],[15,20]],[[20,20],[20,16]],
[[20,20],[15,7]],[[20,17],[7,3]],[[20,23],[7,3]],[[20,17],[12,12]],[[20,23],[12,12]],
"tete"]
from matplotlib.pyplot import *
```

Q1. Tester la commande L[3]. Tester maintenant L[3][0]. Que remarquez-vous ?

Q2. Définir une fonction `tracer(L,e)` qui prend en entrée la liste des segments à tracer ainsi que l'indice associé `e` et qui trace le segment qui est à la position `e` dans `L`.

Q3. Créer une boucle qui parcourt la liste `L` définie ci-dessus et qui trace chacun des segments de cette liste (il s'agit d'un test pour voir si les segments du pendu sont bons !)

Q4. Compléter la boucle précédente afin d'afficher le nombre de segments qui ont été tracés.

Après chaque appel de la fonction tracer, on peut utiliser la fonction `pause(.5)` pour voir à la suite les tracés.
Utiliser plot([X],[Y],'o',markersize=30) pour faire la tête en grand et à la bonne position.

2. ALGORITHME DU JEU

Q5. Définir une fonction `initialisation(mot)` qui prend une chaine de caractère en minuscule `mot` et qui renvoie une liste contenant autant de underscores (« tiret du 8 ») que de lettres à trouver dans le mot.

Vous pouvez de plus afficher un message `"le mot à trouver a ",`nb de lettres du mot`," lettres"`

La fonction `pendu(mot)` va donc avoir le déroulement suivant :

- *On initialise une liste vide à partir d'un mot, et d'autres variables utiles …*
- *On demande une lettre à l'opérateur tant que TOUTES les lettres ne sont pas trouvées et que le nombre de coups joués faux est strictement inférieur à 10.*
- *Pour la lettre proposée, il faut parcourir le mot et remplacer les bonnes lettres dans la liste vide (s'il y en a !). Il ne faut pas oublier de compter combien de lettres ont été trouvées !*
- *Si la lettre n'est pas dans le mot, il faut tracer un des segments du pendu (on utilisera `tracer(L,e)` avec `e` le nb de coups joués faux. Attention à la dernière lettre du mot … et au dixième coup qui est la tête (mettre une condition pour tracer si nb de coups <10) …*

- *Si toutes les lettres ont été trouvées en moins de 10 coups (ou 10 coups) on affiche* `"GAGNE !"`
- *Sinon on affiche la tête du bonhomme et le texte* `text(18,1,"PERDU !")` *ainsi que le mot qui était à trouver.*

Q6. Ecrire cette fonction `pendu(mot)` *et la tester.*

Lexique

Q7. Copier/coller le fichier `Lexique.txt` *dans le même répertoire que votre programme. Faites clic droit et dans propriété copier/coller le chemin de l'emplacement* ***entre les guillemets****. Copier/coller l'ensemble de ces instructions dans votre programme : vous avez maintenant une liste de noms communs dans une liste* `Dictionnaire`.

```
import os
os.chdir("copier/coller l'emplacement ici ! " )

MonFichier=open("Lexique.txt")
Dictionnaire=[]
For Ligne in MonFichier
     Dictionnaire.append(Ligne.strip())
MonFichier.close()
```

SI vous avez ce message d'erreur :

```
SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes in position 2-
3: truncated \UXXXXXXXX escape
```

Remplacer les \ par des /

Mode jeu

Q8. Ecrire un programme qui demande à l'opérateur s'il veut encore jouer (oui/non) et TANT QUE oui, on choisit un mot au hasard dans la liste `tabf` *puis on applique la fonction pendu à ce mot.*

Penser à mettre un `clf()` *pour nettoyer la figure du pendu avant de relancer la boucle et à importer le module* `random` *pour le hasard (importer la fonction randint du module random).*

Q9. Faire 5 parties tout(e) seul(e) ou en famille. Pour chacune de ces parties, modifier la fonction pendu et stocker dans une liste les informations suivantes : Perdu ou gagné, temps mis pour mener la partie à son terme, nombre d'erreurs. Stocker dans une seule et même liste liste_donnees toutes ces données (cette liste doit contenir 15 éléments).

Q10. Définir une fonction tracer_efficacite(liste) qui récupère la liste de la question précédente et trace un graphe avec en abscisse, les numéros des différents essais ; en ordonnée, le temps mis pour jouer la partie. Représenter chaque point du graphe en vert si la partie correspondante est gagnée, et en rouge si la partie est perdue.